Faculdade de Engenharia da Universidade do Porto


# Teste e Validação de um Painel Digital Automóvel usando Visão Computacional

João Pedro Alves Teixeira

VERSÃO FINAL

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Major Automação

Orientador: José Carlos dos Santos Alves
Coorientador: Ricardo Barbosa

Julho 2018

Com o apoio de iTGROW em conjunto com CRITICAL Software, SA

Projeto desenvolvido nas instalações da CRITICAL Software, SA localizada no Largo do Dr. Tito Fontes 21, 4000-060 Porto


# Resumo

Cada vez mais as novas tecnologias estão presentes na nossa vida, sendo que, a cada dia, novos projetos são desenvolvidos com a finalidade de melhorar a vida das populações. A indústria automóvel não é exceção e recorrendo a várias áreas da ciência e tecnologia, áreas de computação, entre outras, desenvolvem mecanismos cada vez mais sofisticados de forma a disponibilizarem ao condutor informação variada a fim de tornar a sua ação facilitada e segura.

Na indústria automóvel, progressivamente existem mais painéis completamente digitais e altamente personalizáveis, que disponibilizam ao condutor variadíssima informação. Alguma desta informação é critica, sendo também aqui apresentada. Aquando da produção destes painéis, existe uma fase de verificação bem como a respetiva validação, para assegurar que tudo está totalmente operacional. Ainda hoje, esta verificação/validação é feita de uma forma manual, através de uma lista onde a pessoa responsável pela validação confirma a resposta do painel aos testes executados. Este processo é muito moroso, tornando-se fundamental o desenvolvimento de um sistema completamente automático. Neste trabalho pretende-se conceber este sistema.

Para isso, existem diferentes fases, a de comando, a de aquisição, a de comparação e a de validação. A fase de comando visa o envio dos comandos teste para o painel e para o *hardware* de aquisição. A interceção das imagens é realizada através de uma FPGA enquanto que a aquisição através de uma câmara. Para a comparação das imagens com as de referência, este projeto faz uso de visão computacional para identificação de símbolos, números, entre outros parâmetros a testar.

Palavras – Chave: Aquisição, Imagens, Processamento, Comparação, Reconhecimento, *framework* de testes.

# Abstract

Technology is a constant presence in our lives and everyday more and more projects are developed to improve life quality.

Automotive industry is not an exception and it is becoming frequent for car panels to be produced completely digital and customizable, which provides the driver with a lot of information. Some of this information is crucial and must be displayed in all circumstances.

When these displays are produced, there is a step to verify and validate them, to assure that they are 100% operational. This process is done manually, going through a check list, where the operator, responsible for the validation, confirms the results of the display tests. This process is too slow, therefore the development of a system where this is done automatically would make sense.

In this project we will develop that system, which contains different stages: the command, the acquisition, the comparison and the validation.

In the command stage, the system sends the test commands to the display and to the hardware responsible for the acquisition.

The acquisition stage consists in two steps, the interception of the image using the Field Programmable Gate Array (FPGA) and the image acquisition from the camera.

For the comparison of the images and the validation of the panel, this project uses computer vision algorithms to identify symbols and numbers, among other patterns.

Keywords: Acquisition, Images, Processing, Comparison, Recognition, test framework.

# Agradecimentos

Em primeiro lugar gostaria de agradecer à CRITICAL Software e à iTGROW, em especial ao coorientador Eng. Ricardo Barbosa e ao Eng. Luís Cruz, por toda a atenção, disponibilidade, ajuda e contribuição neste projeto. Sem eles, não seria possível a realização deste trabalho.

Quero agradecer à Faculdade de Engenharia da Universidade do Porto, pela oportunidade de realizar este projeto, em especial ao Professor orientador José Carlos dos Santos Alves, por ter aceite este desafio de me orientar nesta dissertação, por toda a disponibilidade prestada ao longo do semestre, por todas as orientações para enriquecer este trabalho e por toda a ajuda prestada.

Quero agradecer aos meus pais que tornaram todo este percurso académico possível, que me apoiaram nos momentos mais difíceis e acreditaram que seria capaz.

À minha namorada, Sofia Ribeiro, por toda a atenção, paciência, motivação, força e apoio incondicional que me deu durante esta etapa fundamental do curso.

A todos os meus amigos e colegas de trabalho por de uma forma direta ou indiretamente terem contribuído para o crescimento como pessoa e como estudante e para a finalização deste projeto importante.

A todos muito obrigado...

João Pedro Alves Teixeira

*“The present is theirs;*
*The future, for which I really worked, is mine.”*

*Nikola Tesla*

# Índice

# Lista de figuras

# Lista de tabelas

# Abreviaturas e Símbolos

| | |
|---|---|
| ARM | Advanced RISC Machine |
| AXI | Advanced Extensible Interface |
| CAN | Controller Area Network |
| CCORR | *Cross Correlation* |
| CCOEFF | *Cross Correlation Coefficient* |
| CSW | Critical Software |
| DC | Direct Current |
| DDR | Double Data Rate |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| FMC | FPGA Mezzanine Card |
| FPGA | Field Programmable Gate Array |
| FPS | Frames Per Second |
| GPIO | General Purpose Input/Output |
| HDMI | High Definition Multimedia Interface |
| ID | Identifier |
| IDE | Integrated Development Environment |
| JPEG | Joint Photographic Experts Group |
| JTAG | Joint Test Access Group |
| LVDS | Low Voltage Differential Signaling |
| OCR | Optical Character Recognition |
| PCB | Printed Circuit Board |
| PL | Programmable Logic |
| PS | Processing System |
| RAM | Random Access Memory |
| RGB | Red, Green and Blue |
| SD | Secure Digital |
| SQDIFF | *Square Difference* |
| USB | Universal Serial Bus |
| Zedboard | Zynq Evaluation and Development Boar |

# Capítulo 1 – Introdução

O presente documento é um relatório de estágio realizado na CRITICAL Software em conjunto com a iTGROW e a FEUP.

Neste capítulo encontram-se o contexto, a motivação tecnológica, os objetivos do projeto, a metodologia de trabalho e a estrutura de todo o documento.

## 1.1 - Contexto

A rápida evolução tecnológica levou a que, na industria automóvel, começassem a surgir carros com painéis de instrumentos completamente digitais tal como se pode ver na Figura 1.1[1]. Estes painéis processam toda a informação presente no carro desde a velocidade a que o mesmo circula, a informação relativamente ao consumo de combustível, e/ou energia elétrica, até mesmo à chamada que estamos a receber no telemóvel[2][3].


Figura 1.1 - Exemplo de um painel digital da marca Audi do modelo R8[1].

Tal informação pode ser ou não crítica. Saber qual a estação de rádio que está a tocar no carro é, sem dúvida, uma informação irrelevante do ponto de vista de segurança. Contudo, não nos podemos esquecer que o velocímetro tem que estar a funcionar corretamente; que quando existe um problema no motor a luz acende, em caso de falha; que a informação sobre a quantidade de combustível no veículo tem que estar corretamente assinalada. Estas e outras informações, são certamente críticas para a segurança do condutor. Como tal, existe

uma necessidade extrema de garantir que estes painéis estão devidamente calibrados e operacionais.

Com o mercado automóvel a crescer e cada vez mais exigente, surge a necessidade de criar um sistema de validação dos painéis completamente digitais com as informações vitais do carro. Este sistema de validação tem que ser fiável, rápido, modular, automático e autónomo.

## 1.2 - Motivação

A verificação destes painéis completamente digitais, ou híbridos, é feita através de um colaborador da empresa produtora, utilizando uma lista de verificação, onde os testes são aprovados ou rejeitados. Estes testes passam pelo envio de uma série de comandos para o painel, percebendo assim se este responde como pretendido. Como é de esperar, não são testes cem por cento infalíveis e credíveis uma vez que podem surgir falhas humanas[4].

A monotonia no trabalho pode ser causa para um relaxamento e um desempenho inferior ao normal durante o dia de trabalho de um operador, podendo causar uma falha humana e consequentemente, uma falha na validação do painel[5]. Por esta razão a validação do funcionamento dos painéis digitais através de software e hardware, é importante.

Com base nestes aspetos, surgiu a necessidade de automatizar este processo de validação dos painéis por software. Sendo a CSW produtora de software crítico com alta fiabilidade e qualidade, a entrada neste ramo do ponto de vista da empresa é uma mais valia.

## 1.3 - Objetivos

O principal objetivo deste sistema é poder automatizar o processo de validação de um painel de instrumentos recorrendo a algoritmos de visão computacional. Desta forma, os testes poderiam ser executados a qualquer momento e de forma completamente automática, garantindo assim que as validações são o mais fiáveis possíveis, uma vez que não requeria intervenção humana, que pode originar falsas validações. Para realizar este projeto será necessário a criação de uma *framework* de testes e de uma unidade de aquisição para capturar as imagens do painel de teste.

Um outro objetivo deste projeto é poder realizar esta validação o mais rápido possível de forma modular para poder integrar mais tarde com outros sistemas.

## 1.4 - Metodologia de trabalho

Em termos de desenvolvimento de software, existem duas grandes metodologias "*Waterfall*" e "*Agile*", sendo a última a usada durante este projeto, uma vez que é a adotada pela empresa nos projetos relacionados.

As fases de elaboração de software estão divididas em:

- Requisitos – É nesta etapa que todos as exigências ficam acordadas, isto é, tudo o que é suposto o software cumprir fica documentado. O documento de arranque desta fase, por norma, é um contrato ou proposta de trabalho. O que vai ser produzido no final, é o documento de requisitos.
- Planeamento – Utilizando o documento de requisitos como entrada para esta nova fase, segue-se o planeamento de todos os recursos humanos e materiais, que irão ser necessários para o desenvolvimento do projeto. No fim desta fase é construído um documento contendo toda essa informação.
- Desenvolvimento – Nesta fase há o desenvolvimento do software. No caso da metodologia "*Waterfall*", esta é a fase que poderá ser a mais longa.
- Verificação e validação – Após o desenvolvimento, existe um período de testes, verificação, correção e validação para diminuir a probabilidade da existência de "*bugs*". No fim desta fase normalmente o projeto termina e inicia-se a próxima fase por tempo indeterminado.
- Manutenção – Apesar de o produto já ter sido entregue, é importante ir aperfeiçoando-o, uma vez que podem surgir melhorias aplicadas ao software e a necessidade de responder aos novos requisitos do cliente.

Caracterizando estas metodologias de trabalho, a primeira, assenta num método de trabalho mais rentável quando falamos em projetos fixos em que nada ao longo do projeto varia tanto a nível de tempo de desenvolvimento bem como a nível de custo do projeto.

Já na segunda metodologia de trabalho, "*Agile*", existem muitas variantes da mesma, inclusive uma combinação entre as duas metodologias. No entanto, esta tem todos as fases de desenvolvimento de *software*, mas de uma forma interativa. Os requisitos vão surgindo ao longo do projeto. As funcionalidades, implementadas ao mesmo tempo e testadas logo de seguida, mantendo uma interação constante com o cliente, mostrando resultados ao longo do tempo[6][7].

Para este projeto, foram planeadas reuniões semanais com o orientador da CSW, para poder definir os requisitos e para haver uma apresentação do trabalho realizado ao longo do semestre. As funcionalidades do trabalho desenvolvido, foram testadas ao longo da sua implementação, sendo depois apresentadas ao orientador na empresa.


Figura 1.2 - Diagrama Metodologia Agile[8]

Relativamente ao projeto, este foi organizado de acordo com as diferentes unidades. Começou-se por estabelecer a comunicação com o painel de instrumentos, depois começou-se por desenvolver a unidade de processamento e os algoritmos para comparação das imagens seguidos da unidade de aquisição. Aqui foram realizados, à medida que foram desenvolvidas, os devidos testes para cada uma das unidades. Por último a unidade de interceção necessitaria de modificações bem como a produção de um circuito impresso. A última fase seria a de integração de todas as unidades e testes finais. Na realização do projeto houve alguns desvios do planeamento já que devido a problemas técnicos com o painel, não foi possível testa-lo.

## 1.5 - Estrutura

A estrutura da dissertação, está dividida em 6 capítulos. O Capítulo 1 contém a introdução onde se descreve o contexto, a motivação tecnológica, a metodologia do trabalho e os objetivos desta dissertação.

No Capítulo 2, encontra-se toda a pesquisa sobre o trabalho existente no mercado bem como os métodos também já existentes para a comparação das imagens. Também neste capítulo está incluído todo o estudo do trabalho realizado anteriormente, no âmbito deste projeto.

No Capítulo 3, estão escritas informações dos métodos usados na unidade de processamento para a validação do painel através da imagem capturada. Neste capítulo,

vamos perceber um pouco no que consiste a unidade de processamento, como está organizada e a sua interface com o utilizador.

No Capítulo 4, podemos ler mais informações sobre a unidade de aquisição e o que foi desenvolvido relativamente a esta.

No Capítulo 5, estão presentes os resultados dos métodos aplicados para a validação das imagens, bem como, tempos de processamento relativos à aquisição e processamento. No fim do capítulo podemos encontrar todo o trabalho desenvolvido na unidade de interceção.

No último capítulo, o Capítulo 6, encontra-se as conclusões e o trabalho futuro. Após este capítulo podem-se encontrar os anexos e as referências bibliográficas.

# Capítulo 2 – Estado de Arte

Num momento inicial, dada a dimensão do projeto global, tornou-se crucial compreender como era o funcionamento do painel a testar, seguido da comunicação entre o computador e o painel de teste. Da mesma forma, foi importante depreender o que foi realizado sobre este projeto nos anos anteriores, bem como pesquisar e perceber mais sobre métodos de comparação de imagens e reconhecimento de símbolos.

## 2.1 - Painel de teste

Apesar de estar definido a forma como irá ser testado e validado o painel, foi necessário fazer uma análise de como é feita a validação dos painéis de instrumentos hoje no mercado. Para a marca de automóveis Audi[9], a verificação do painel de instrumentos do modelo TT, passa por uma bancada de testes em que são testados todos os sistemas eletrónicos. Como se pode ver na Figura 2.1, é usada uma câmara para a aquisição das múltiplas configurações, que este painel pode adotar, para posteriormente ser feita a validação. Como é de esperar não é só o painel de instrumentos principal que é testado, mas também botões que contêm a informação digital. Outros sistemas para além de usarem sistemas de validação com o recurso a visão computacional, também fazem uso de um "*hardware-in-the-loop*"[10], tal como se pretende reproduzir neste trabalho.


Figura 2.1 - Bancada de testes da Audi para o painel do modelo TT[9].

Para que a análise ficasse completa, foi essencial obter mais informações sobre o painel a testar. O *cluster*, ou painel de instrumentos, é constituído por 2 partes. A parte do controlo e processamento de toda a informação presente no painel, e a parte da apresentação da imagem, neste caso, através de um ecrã. Na parte de controlo e processamento, podem-se identificar alguns circuitos integrados, fitas de ligação com o ecrã, o altifalante para alertas sonoros, o conector principal do painel com o exterior e ainda um conector LVDS, tal como descrito na dissertação[11]. Relativamente ao ecrã, este apresenta uma resolução de 1280 pixéis horizontais e 480 *pixéis* verticais, com uma proporção de 8:3 e uma dimensão de 12.3'' diagonal de ecrã. Devido a questões de confidencialidade e/ou segredo industrial, não foi possível conseguir mais informações relativas ao painel, pelo que, todas estes dados foram obtidos de uma forma visual ou através da escassa documentação que foi fornecida.

Figura 2.2 - Visão frontal do painel de instrumentos de teste.

Figura 2.3 - Visão traseira do painel de instrumentos de teste.

Para este projeto, é relevante a compreensão de todo o painel, destacando-se especialmente as ligações exteriores e ligação painel-ecrã, já que esta contém a informação a intercetar. Por último era necessário estudar a comunicação entre o painel e a unidade de aquisição. Esta comunicação é feita com base num protocolo CAN.

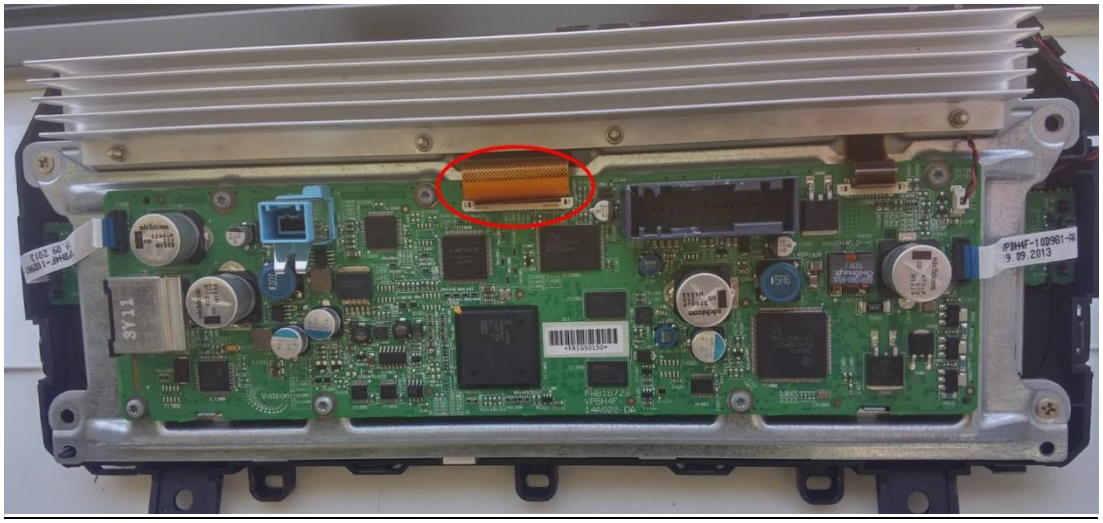


Figura 2.4 - Visão do interior do painel. Assinalado na imagem está a fita com a informação a intercetar.

## 2.2 - Protocolo CAN

O protocolo CAN é um protocolo usado para mensagens curtas que podem transportar até 8 bytes de dados, tornando-o ideal para transportar mensagens de *trigger* e valores medidos[12]. Neste protocolo cada mensagem apresenta uma prioridade para questões de acesso, caso hajam várias transmissões em simultâneo. Quando se pretende transmitir uma mensagem, tem que se esperar que o barramento esteja desimpedido para que esta seja enviada. Relativamente às *frames* CAN, estas apresentam um formato bem definido para que o recetor as possa interpretar. Assim, existem dois tipos de formatos, ambos representativos da versão 2.0 do protocolo CAN, as *frames* de tamanho standard e as de tamanho estendido. A diferença entre elas encontra-se no tamanho dos bits identificadores que no primeiro caso são de 11 bits e no segundo caso são de 29 bits. A mensagem CAN inicia-se sempre com um bit denominado por *Start Of Frame* (SOF) que serve para o recetor identificar o início da mesma. Para o formato Standard, depois do primeiro bit, existe o campo de arbitragem e o campo de controlo. O campo de arbitragem é constituído pelos bits do identificador e um bit de controlo denominado *Remote Transmit Request* (RTR). Este bit permite ao recetor distinguir entre *frames* de dados e *frames* remotas sendo o bit chamado "dominante" para *frames* de dados e "recessivo" para as remotas. O campo de controlo, apresenta dois bits para a identificação do tipo de *frame*, seguido do *Data Length Code* (DLC). A seguir encontram-se os dados a serem transmitidos, outro bit para confirmação e um último indicador de fim de trama, mais conhecido como *End Of Frame* (EOF)[13]. Na Figura 2.5 está exemplificada a mensagem CAN versão 2.0 *Standard*.

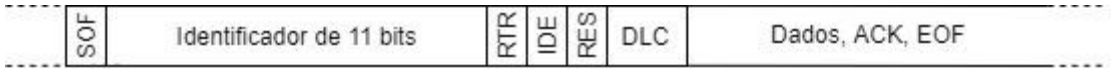



Figura 2.5 - *Frame* CAN versão 2.0 *Standard* com os respetivos bits. IDE representa o *Identifier Extension* bit e RES o chamado bit reservado.

Para a comunicação entre o computador e o painel, através do protocolo CAN, tornou-se necessário instalar uma interface capaz de fazer tal comunicação. Para isso foi usado uma ferramenta da PEAK[14] que faz a ligação entre o painel e o computador por *usb*. Depois de instalados os *drivers* referentes a esta ferramenta, houve necessidade de instalar também, uma ferramenta que pudesse estabelecer a comunicação. Para esta comunicação, utilizou-se o "*PCan view*[1]" que é um *software* bastante intuitivo e de fácil utilização.

## 2.3 - Trabalho FPGA

O processamento de imagens também pode ser feito com recurso a uma FPGA e não só com uma câmara como é exemplificado no trabalho[15]. Em diversos projetos verifica-se o uso da FPGA como complemento de um sistema com o objetivo de o tornar mais rápido. Outros trabalhos usam exclusivamente a FPGA para a captura e para a comparação das imagens. A FPGA oferece não só poder de processamento de vídeo, mas também, é uma versão mais económica e compacta, face a outras soluções, comparadas com a velocidade de processamento[16][17].

Para melhor compreensão do que já existia relativo a este projeto, analisou-se todo o trabalho que tinha sido criado anteriormente. A solução para o desenvolvimento da unidade de interceção, escolhida nos trabalhos anteriores[11][18], foi a *Zedboard*. Esta apresenta um processador Zynq-7000, *Dual* ARM *Cortex*–A9 *MPCore* que chega até aos 667MHz. Relativamente à memoria, apresenta 512 MB DDR3 e flash de 256 Mb, mais que suficiente para a função que iria desempenhar. Relativamente à expansibilidade, a *Zedboard* apresenta uma interface FMC com terminais suficientes para uma interceção da imagem do painel, no entanto apresentava alguns terminais internos referentes a tensões pelo que nem todos poderiam ser utilizados. Ainda assim era uma ficha mais que suficiente para utilizar neste projeto.

A *Zedboard*, é alimentada por 12V DC com máxima corrente de 3 amperes e apresenta uma porta USB JTAG para ser reprogramada[19]. As outras especificações podem-se encontrar na hiperligação para o website da *Zedboard*, presente nas referências bibliográficas.

A *Zedboard* permite a utilização de um sistema Linux no seu processador tendo sido escolhido o *Petalinux* no projeto anterior[18]. Existiam outras possibilidades como o *Linaro*[2], no entanto, o facto do *Petalinux* ser gratuito e apresentar diversas ferramentas de suporte,

[1] https://www.peak-system.com/PCAN-View.242.0.html?&L=1

[2] https://www.linaro.org/

contribui para que a escolha fosse esta, face a todos os outros sistemas pesquisados. O código em C escrito e a ser executado no *Petalinux*, teve que ser estudado. Este assenta numa comunicação por *sockets*, onde espera uma ligação por parte de um cliente, ligado por *Ethernet*. De seguida, faz uso dos sinais obtidos e reconstrói a imagem intercetada enviando-a de volta para o cliente. Para que todo este processo pudesse ser testado, foi necessária uma configuração dos *jumpers* da *Zedboard* para ser inicializada a partir do cartão SD. Depois de inicializada, esta requer um utilizador e uma palavra chave para que o sistema do *Petalinux* inicialize, tal como se pode verificar pela imagem do anexo C. Após esta inicialização é necessária uma outra configuração da placa de rede permitindo assim, que o sistema seja capaz de inicializar o programa, ficando à espera então da comunicação por parte do cliente.

O Vivado da Xilinx é uma das ferramentas mais utilizadas na visualização e programação de *FPGA's*. Este software é de fácil utilização e uma vez que foi utilizado em anos anteriores, seria uma ferramenta essencial para melhor compreensão daquilo que foi feito anteriormente. Uma análise do *datapath* configurado na *Zedboard*, permitiu perceber como era reconstruída a imagem e como é que este *datapath* era constituído.

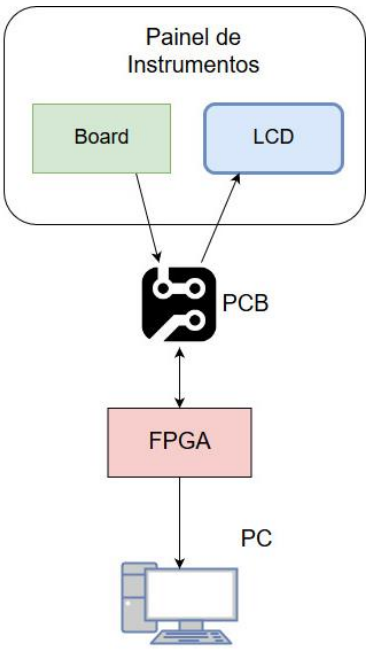



Figura 2.6 - Ilustração do projeto anterior com a adição da PCB desenvolvida nesta dissertação

Após a análise completa do sistema criado em anos anteriores, relativo a este projeto, e para que a interceção da imagem do *cluster* fosse possível, ficava a faltar a criação de uma PCB (*Printed Circuit Board*), que enviaria os sinais que recebia do *cluster* para a *Zedboard*. Também sobre este tema, seria necessário estudar como seria possível o desenho desta placa e que boas praticas deveriam ser adotadas para a construção da mesma, dado que este tema nunca foi abordado durante a formação académica. Chegou-se à conclusão, com a ajuda de

alguns elementos da empresa onde o projeto iria ser desenvolvido, que se deveria manter uma proximidade entre os pares de sinais e ainda tentar ao máximo, eliminar as vias dos terminais dos componentes a serem assemblados na placa. Ainda sobre o estudo dos sinais que iriam passar na PCB, não seria necessário a inclusão de nenhum componente eletrónico na mesma, para além das fichas de fariam a ligação entre os sistemas.

Relativamente às ferramentas possíveis para o desenho desta PCB, foram encontradas algumas como "EasyEDA[3]", "Kicad[4]", "Allegro[5]", "PROTEL[6]", "TinyCAD[7]", "ExpressPCB[8]", "Eagle[9]" e ainda "PCBWeb Deigner[10]" sendo este último desenvolvido online. Depois de um teste a todos eles, foi escolhido o "Eagle" dada a fácil utilização.

Uma vez que esta placa não poderia ser desenvolvida na CRITICAL Software, teve que se efetuar um estudo do mercado relativo as empresas produtoras de *PCB's*. A Guimocircuito[11] foi a única empresa que fazia a produção da placa assemblagem dos componentes e teste dos mesmos, cujo orçamento estaria dentro do esperado por parte da CSW para este projeto.

## 2.4 - Métodos de comparação de imagens

Todos os dias, existe um aumento do número de aplicações em que visão computacional é usado, sem sequer nos apercebermos que a estamos a utilizar[20]. Aplicações tais como: o reconhecimento de letras, números e faces humanas; na indústria produtora, a inspeção de máquinas produtoras para deteção de defeitos; a modelação de objetos 3D em imagens capturadas; deteção de doenças em variadíssimas partes do corpo neste caso no ramo da medicina; contagem de objetos em movimento e reconhecimento dos mesmos; reconhecimento de impressões digitais; e ainda no ramo automóvel, para deteção de sinais de trânsito, pessoas na proximidade da viatura em perigo, ou mesmo, como é o caso deste trabalho, reconhecimento de falhas no hardware e software[21][22][23].

[3] https://easyeda.com/pt
[4] http://kicad-pcb.org/
[5] https://www.cadence.com/content/cadence-www/global/en_US/home/tools/pcb-design-and-analysis/pcb-layout/allegro-pcb-designer.html
[6] https://www.altium.com/solution/protel-pcb-design
[7] https://www.tinycad.net/
[8] https://www.expresspcb.com/
[9] https://www.autodesk.com/products/eagle/overview
[10] http://www.pcbweb.com/
[11] http://www.guimocircuito.com/

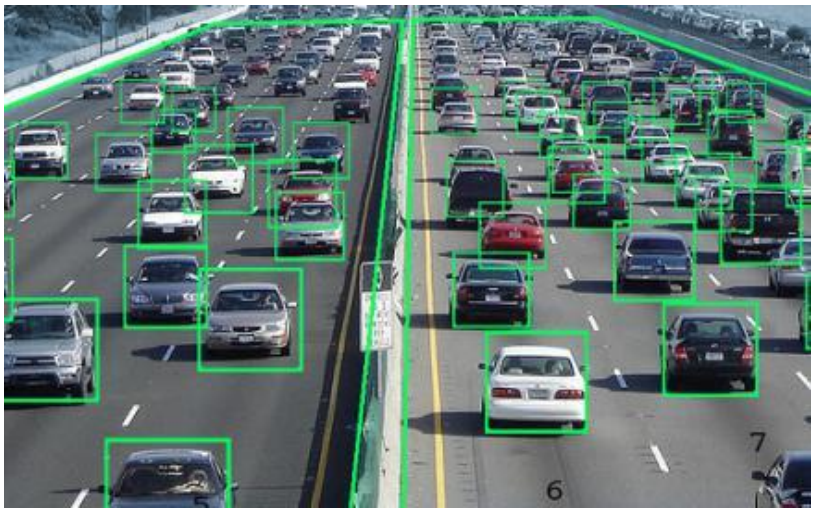


Figura 2.7 - Exemplo de reconhecimento de veículos numa autoestrada[24]

Tal como referido no trabalho[25], nas últimas 5 décadas o mundo sofreu um avanço gigantesco no que diz respeito à manipulação e processamento de imagens, facto este devido à melhoria continua dos sistemas informáticos e tecnológicos. Maior espaço de armazenamento e maior capacidade de processamento das imagens, são sem dúvida os principais requisitos necessários para desenvolvimento na área de manipulação e processamento das mesmas.

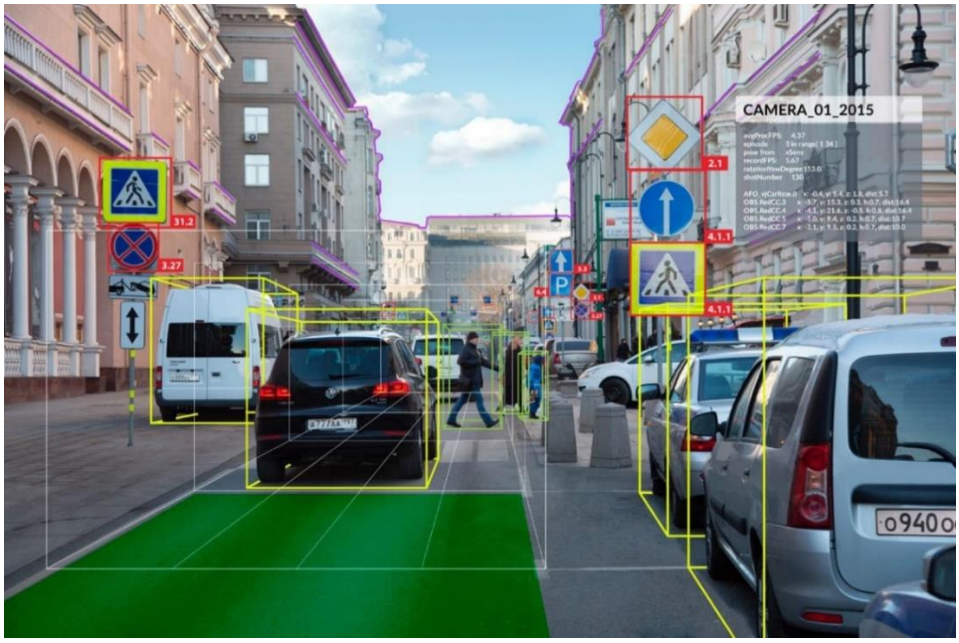



Figura 2.8 - Exemplo de reconhecimento de sinais de transito e veículos[26]

Antes de avançar para a criação de um sistema de reconhecimento de imagens deve-se compreender alguns aspetos da visão humana, isto porque, dada a forma como normalmente é feita a validação de painéis digitais, alterações mínimas podem não ser percetíveis. A visão humana é sensível à luz na região do espectro eletromagnético, entre o comprimento de onda

de 350nm aos 780nm, e capaz de identificar 3 propriedades da luz: a cor, a saturação e o brilho, que são as principais variantes numa imagem.

Quando se fala em processar imagens e manipulá-las, falamos em captura e compressão para transmissão, falamos em reconhecimento de padrões no processamento das imagens, falamos em "*machine learning*", e falamos também em reconstrução de imagens.

Para este projeto presume-se que o ambiente de testes seja controlado a nível de luz uma vez que a aquisição e posterior processamento depende desse fator. Ainda assim, em casos que isso não é possível, existe um outro método para melhorar o brilho e contraste das imagens a testar, que se chama "equalização do histograma". Este método serve matematicamente para normalizar a frequência de ocorrência dos valores das componentes da imagem[27]. Assim, existe o histograma da imagem[21] que indica a quantidade de cores na mesma, sendo possível manipulá-lo, dividindo a imagem em alguns blocos e aplicando a fórmula seguinte depois de simplificada:

$$C(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = C(I-1) + \frac{1}{N}(h(I)) \qquad (2.1)$$

N representa o número de pixéis na imagem, h(i) o histograma original da imagem e C(I) o novo histograma. Esta pequena manipulação, solucionaria problemas de deteção dos símbolos referentes ao brilho da imagem. Também para este projeto, são usados os dois círculos dos mostradores como referência para calibração, e se a imagem estiver trabalhada, é facilmente percetível estes dois círculos de maneira a que a calibração seja 100% correta.

Falando um pouco sobre compressão para transmissão e armazenamento, alguns trabalhos[25] defendem que o objetivo nesta área é representar a imagem com o menor número de bits possível. Reduzindo o número bits, pode causar alguma distorção em relação à imagem original, no entanto consegue-se alguma capacidade para comprimir a imagem facilitando o armazenamento e a transmissão, sendo o ponto ótimo, comprimir a imagem o mais possível, até ao ponto em que causa menor distorção capaz de alterar a imagem significativamente. Idealmente, poderemos querer saber qual o rácio entre compressão/distorção, para o caso que estamos a trabalhar neste projeto.


Figura 2.9 - Exemplo de imagem sem compressão.

Figura 2.10 - Exemplo de imagem com cerca de 88% de compressão JPEG.

Para a compressão das imagens, o método mais usado é o JPEG (*Joint Photographic Experts Group*). Este método foi criado por um grupo formado em 1986 que mais tarde em 1993, publicou um standard. A primeira parte deste standard já apresentava um conjunto de técnicas para compressão de imagens[28]. A compressão começa na utilização da imagem original em RGB, e convertendo-a para YCbCr, que é uma representação da imagem em espaços de cores. Depois disso são aplicadas equações e transformadas de forma a minimizar as perdas e aumentar a compressão.

Tal como para compressão de imagens, para vídeo também existe um tipo de compressão chamada "H.264 *Advanced Video Coding*"[29]. Este tipo de compressão de vídeo, é muito usado em sistemas de videovigilância, já que existe uma necessidade de armazenar a maior quantidade de vídeo possível. Uma vez que este tipo de compressão adiciona perdas de informação, pode não ser a solução ótima quando usada para efeitos de reconhecimento facial.

Relativamente ao reconhecimento de padrões, existem várias formas de o fazer. Deteção de bordas ou contornos também mais conhecido como "*Edge Detection*", é muito usado nas

imagens para reconhecimento de formas. Este método assenta na transição abrupta de regiões de cinza.

Ainda para deteção de formas, as transformadas de *Hough* são muito usadas tal como é explicado nos trabalhos[30][31]. Esta técnica surgiu inicialmente para a deteção de linhas na imagem, e foi mais tarde aperfeiçoada e não só detetava linhas como também círculos. Ao longo dos anos passou a conseguir identificar outras formas mais irregulares inclusive imagens 3D. Ainda assim, esta transformada apresenta algumas limitações como é natural. Uma delas passa pela qualidade da imagem a analisar. Quanto pior a qualidade da imagem, pior a eficácia da transformada de *Hough*. Além disso a eficiência desta transformada varia muito com o "ruido" de background. Apesar de tudo, a transformada de *Hough* continua a ser uma das transformadas mais usadas no que diz respeito a identificação de formas e reconhecimento em imagens digitais. A imagem a analisar deverá ser minimamente tratada antes de ser aplicada a transformada de *Hough*[32].

Outro método também muito usado em reconhecimento de objetos, é a deteção de cantos ou, "*corner detection*". Um canto pode ser definido pela intersecção de duas pontas de retas. Um "bom" canto é aquele que tem uma posição bem definida na imagem e consegue ser facilmente identificado. Sobre este método existem muitos algoritmos relacionados, dando destaque a um dos primeiros e mais simples, mas computacionalmente pesados o algoritmo de "*Moravec*"[33]. Fazendo uma breve descrição do funcionamento do mesmo, este algoritmo testa todos os pixéis da imagem, procurando em cada pixel da sua vizinhança uma certa semelhança calculada a partir da soma da diferença dos quadrados. Se a vizinhança for igual ao pixel central, então o resultado do cálculo vai ser elevado, logo o algoritmo não deteta nenhum canto. Se a vizinhança tiver um número definido de pixéis diferentes, então estamos perante um canto. Este método apresenta uma maior eficácia, depois de à imagem ter sido aplicado um outro método dos contornos.

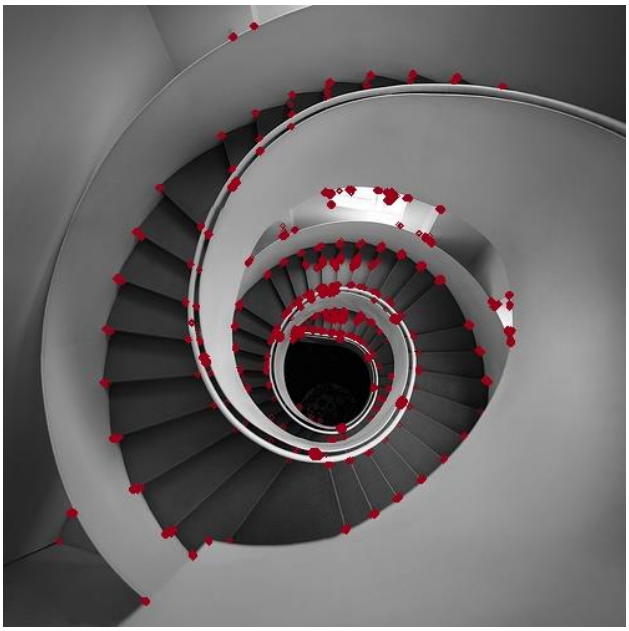


Figura 2.11 - Exemplo de deteção de pontas numa imagem com o recurso ao *OpenCV*[34]

Quando se pretende comparar duas imagens, existem varias questões relacionadas com a comparação, no entanto a que mais sobressai, é sem duvida, quanto e que se aceita que duas imagens são iguais[35]. Para responder, pode-se muito facilmente fazer a comparação entre duas imagens, subtraindo uma a outra, obtendo assim um valor que seria 0 se elas forem iguais. Este método de resolução, é muito usado, no entanto com uma condicionante de que a câmara tem que estar sujeita sempre as condições de luz e na mesma posição, caso contrário, pequenos desvios nesse sentido, podem fazer com que haja uma diferença de pixéis enorme, rejeitando assim a possibilidade de as imagens serem iguais, quando realmente são. Estas pequenas diferenças não são percetíveis a olho humano dai, só em casos muito específicos e controlados é que se usa este método.

Figura 2.12 - Exemplo da utilização do método *matching template*[36]. Na imagem da esquerda podemos observar o resultado dando destaque ao ponto preto já que representa o ponto ótimo para a identificação do *template* que neste caso é a cara do jogador.

Surge então outro método de comparação bastante usado no dia a dia por exemplo em reconhecimento de sinais de transito nos carros mais modernos, de nome "*Template*

*matching*". Este método usa uma imagem, *template*, mais pequena que a imagem que se vai comparar, e percorre essa imagem do início ao fim para poder encontrar a melhor correspondência da diferença entre elas. Apesar deste método ser relativamente simples, existe ainda uma certa complexidade porque mesmo aqui, existem problemas em saber quando se aceita o resultado, isto é, qual o valor mínimo para dizer que existe um *template* naquela imagem. Existe então uma técnica que é saber a região entre a imagem original da imagem do *template*. Esta região pode ser calculada de diferentes formas, entre elas destaca-se a soma das diferenças absolutas, a máxima diferença entre as distâncias e a soma do quadrado das distâncias. Mais uma vez, este cálculo pode muitas vezes induzir em erro e por isso passaram a existir as versões normalizadas passando neste caso o intervalo a estar compreendido entre 0 e 1 sendo que 1 obtinha o valor de máxima proximidade e 0 o valor sem proximidade nenhuma. Outro método de cálculo da distância é através do "*Chamfer Matching*". Este algoritmo oferece uma forma mais eficiente de aproximar a distância para imagens binarias. Este método tem por base os valores acumulados da distância.

Para este projeto, foi necessário pesquisar e definir quais as ferramentas de hardware e software a serem usadas. Entre as quais podemos destacar o *CodeBlocks*[12], sendo o principal motivo da seleção e utilização o facto de este ser gratuito e bastante intuitivo. Este IDE esta orientado para linguagens de programação como o C ou C++, que foram duas das linguagens que poderiam ser usadas para o desenvolvimento de todo o projeto.

Ainda relativamente às ferramentas disponíveis para comparação de imagens, o *OpenCV* destaca-se face às outras. O *OpenCV* foi construído para aplicações de visão computacional, mas também para *machine learning*[37]. Este software representa uma biblioteca muito completa, onde utiliza mais de 2500 algoritmos que servem para identificação de caras, objetos, classificar ações de pessoas em vídeos, encontrar imagens numa base de dados com um grau de semelhança, entre muitas mais funcionalidades[38]. Adicionalmente o suporte que esta biblioteca possui por parte da comunidade de utilizadores, facilita o esclarecimento de eventuais duvidas que possam surgir aquando da utilização das funções, fazendo com que o seu destaque seja notório. Esta biblioteca é usada em empresas como a *Google*, *Yahoo*, *Microsoft*, *Toyota*, *Sony*, *Honda*, e outras *startups*, que também reconhecem o potencial deste software, nos seus projetos. Setores como a indústria também beneficiam com o uso desta biblioteca para os seus robots, para que estes possam reconhecer caixas ou símbolos que sejam importantes na assemblagem, seleção ou organização de um armazém, por exemplo.

[12] http://www.codeblocks.org/

# Capítulo 3 – Validação do painel pela análise da imagem de teste

Neste capítulo, será abordado o tipo de testes que devem existir fazendo uma breve explicação do seu funcionamento e o porque da escolha desses. Será abordado a importância de existir uma sincronização de relógios entre os sistemas, bem como, explicado a importância dos comandos de captura, os de configuração e o porquê da utilização destes comandos específicos. Por último vai ser feita referência à forma como o utilizador interage com o sistema.

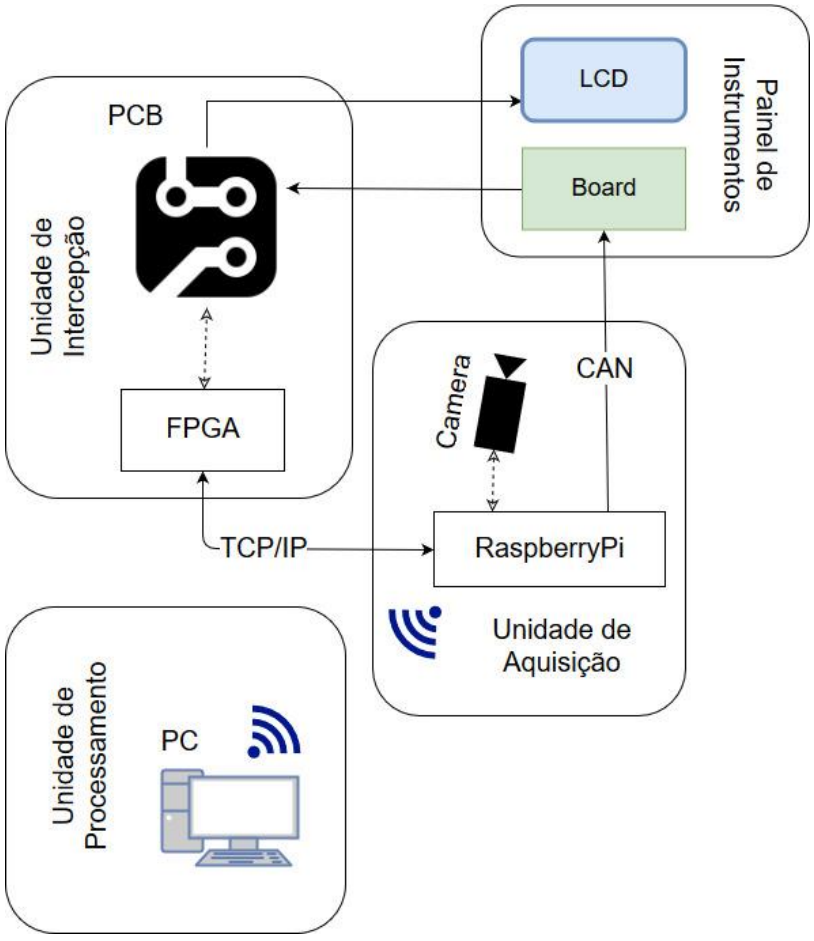



Figura 3.1 - Visão global do sistema. A tracejado encontramos comunicação entre os vários sistemas na mesma unidade.

Na Figura 3.1 podemos perceber a comunicação entre as diferentes unidades presentes no projeto. Neste capítulo iremos falar sobre a unidade de processamento e a comunicação da mesma com a unidade de aquisição.

## 3.1 - Tipos de testes

Antes de desenvolver o algoritmo de execução de testes, houve a necessidade de definir que tipo de testes tinham de ser realizados.

Após a primeira reunião com o orientador da CSW, foram decididos os casos de uso neste projeto. Para validar este painel, após a calibração, teriam que ser efetuados testes a símbolos, a números, as barras horizontais e verticais e aos mostradores da velocidade e das rotações do carro.

Com estes casos conseguiu-se identificar pelo menos 3 tipos de testes. Os testes aos símbolos e aos números, os testes de *sweep,* que assentavam na captura de vídeo em vez de imagem estática, para que pudesse ser decomposto em várias *frames* e comparadas, garantindo assim que os mostradores apresentavam uma variação contínua e não uma variação aleatória.

Para além disso, foi identificado um primeiro teste, que serviria para calibrar a imagem obtida face à referência. Esta calibração poderia ser horizontal ou vertical, mas também, angular, caso a imagem capturada tivesse alguma rotação face a posição de referência. Para qualquer caso, a *framework* deverá identificar a deslocação da imagem capturada face a imagem original e aplicar as deslocações necessárias para que ambas estejam alinhadas aquando da comparação.

Para este projeto, também foram feitos testes de *performance*, apesar de não haver termo de comparação, o objetivo seria perceber quanto tempo demoraria a aquisição da imagem do painel, o teste e a validação do mesmo.

## 3.2 - Testes de símbolos

Para este teste, foi usada uma função do *OpenCV,* designada de *Template Matching*[39]. Esta função, procura uma imagem *template*, na imagem original, através de um método *sliding* que percorre a imagem toda. Este método, pode usar um dos 6 parâmetros disponíveis. Estes parâmetros são usados como métodos de paragem e aceitação da região onde o *template* é encontrado.

Apesar do método ter se revelado um sucesso no reconhecimento do *template* na imagem, como se pode ver no capítulo dos resultados, quando existia dois símbolos iguais na imagem, a função retornava o primeiro que encontrava. Relativamente a cores, a função

também se tornava limitada uma vez que no seu processamento, a imagem original e a imagem do *template*, eram transformadas em escalas de cinza.

### 3.2.1 - Métodos para identificação do *template*

No método SQDIFF, aquilo que acontece é que o resultado, é calculado pelo somatório do quadrado da diferença entre a posição do *template* menos a posição da imagem original menos a do *template*. Uma vez que estamos a falar de uma diferença entre dois resultados em x e y, o valor mínimo vai ser a posição pretendida. Segue a fórmula matemática para o cálculo da região em que R é a imagem resultante da comparação, I a imagem original e T a imagem do *template* a ser pesquisado:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2, \tag{3.1}$$

O método SQDIFF_NORMED, usa a expressão do método SQDIFF e normaliza-o tal como o nome indica. Segue a expressão referente ao método SQDIFF_NORMED:

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}}, \tag{3.2}$$

À semelhança do método anterior, o método CCORR, também apresenta uma versão normalizada. Neste caso, a operação difere, em relação a (3.1), em vez de ser a diferença entre os pontos, é um produto. Conseguimos perceber com mais detalhe este método através da fórmula matemática seguinte em que mais uma vez R é a imagem resultante da comparação, I a imagem original e T a imagem do *template* a ser pesquisado:

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')), \tag{3.3}$$

Também este método, novamente em relação ao método anterior, apresenta uma versão normalizada, CCORR_NORMED. Por norma esta versão normalizada, e com base em testes feitos a várias imagens, apresenta resultados mais fidedignos, tal como era esperado que tal acontecesse dada a normalização. Assim segue a expressão do método em estudo, normalizada.

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'}(T(x',y'))^2 \cdot \sum_{x',y'}(I(x+x',y+y'))^2}}, \tag{3.4}$$

Este último método, constituído por CCOEFF e a sua versão normalizada, é do ponto de vista matemático, o mais complexo de todos, isto porque neste método, não são usados diretamente a imagem do *template* e da imagem original, mas sim uma transformação das mesmas. A transformação do *template*, passa por uma diferença entre a imagem do *template* e o produto de um somatório por uma razão que envolve a altura e largura da mesma imagem. O mesmo acontece para a imagem original, tal como se pode ver nas expressões seguintes em que R é a imagem resultante da comparação, I a imagem original, T a imagem do *template* a ser pesquisado, *w* a largura e *h* a altura da imagem original:

$$T'(x',y') = T(x',y') - \frac{1}{(w.h)} \cdot \sum_{x'',y''} T(x'',y''), \tag{3.5}$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{(w.h)} \cdot \sum_{x'',y''} I(x+x'',y+y''), \tag{3.6}$$

Assim a expressão do método em questão surge na expressão (3.7) e a versão normalizada podemos ver na expressão (3.8), tal como nos outros métodos, com melhores resultados face a versão não normalizada.

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))^2, \tag{3.7}$$

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}, \tag{3.8}$$

## 3.3 - Testes de Números

Para este teste, o objetivo era a identificação de números no painel conforme o teste o indicasse. Após uma vasta pesquisa, conseguiu-se perceber que para podermos encontrar determinados números na imagem, poder-se-ia usar o mesmo método. A primeira baseava-se no método anterior, e tendo na nossa base de dados um *template* para cada dígito, iriamos fazer um teste dígito a dígito na região até encontrarmos um resultado ótimo. Caso esse dígito fosse o desejado então, passaríamos para o dígito seguinte, com a indicação de que o anterior estava correto. Caso contrário, teríamos a indicação oposta, isto é, que o dígito anterior estaria errado, terminando aqui a validação do número. Repetiríamos o processo até chegarmos ao fim do número completo, apresentando de seguida, os resultados negativos,

caso falhasse algum teste, e positivos caso todos os testes estejam de acordo com o esperado.

## 3.4 - Testes de *Sweep* para indicadores dinâmicos

Este tipo de teste é usado para movimento angular ou linear de indicadores, tal como, o ponteiro de um mostrador de velocidade ou um indicador da medida de combustível presente no painel de teste. Na produção de um painel completamente digital, a variação de velocidade apresentada sob a forma de uma imagem em tudo semelhante a um mostrador convencional, deve ser o mais fluida possível, para que, não haja pequenas oscilações de velocidade durante essa variação. Para este caso, teria que ser avaliado então, a forma como varia o ponteiro digital.

Para testar tal fluidez, pensou-se em criar um vídeo e analisar este vídeo. O requisito imposto para este tipo de testes, não falava no mínimo de *frames* por segundo que o vídeo deveria ter, no entanto tomamos como referência um objetivo de 60 *frames* por segundo, aquando do desenvolvimento do projeto.

Para poder fazer a comparação e a análise do vídeo, este teve que ser decomposto em várias *frames*. Desta forma era possível a comparação de *frame* a *frame*, verificando assim se todos os momentos estavam de acordo com o que era esperado. Para além desta verificação era feita uma outra comparação tanto à primeira *frame*, que seria o momento imediatamente antes do ponteiro se mexer, e à última *frame*, que seria o momento imediatamente depois do ponteiro parar. Fazendo a comparação das imagens entre si conseguiríamos saber o ângulo entre o ponteiro da primeira imagem e o ponteiro da última imagem. Este angulo teria que ser igual ao esperado. Para finalizar é feita, uma última verificação comparando a posição do pixel do ponteiro mais longe do centro do mostrador tanto do velocímetro como do tacómetro. Como critério de aceitação, foi estabelecido um desvio da posição esperada, de 1 grau angular e de 0,5% do tamanho horizontal do painel em pixéis para o desvio horizontal e 0,5% do tamanho vertical do painel em pixéis para o desvio vertical fazendo assim um desvio de cerca de 6 pixéis e 2 pixéis horizontal e vertical respetivamente. Todos os desvios com diferença superior a estes valores, teriam que ser rejeitados e analisados a posteriori. Estes valores foram obtidos com base na imagem utilizada para os testes, no entanto no caso de captura de uma imagem do painel estes valores teriam que ser novamente ajustados.

Antes de ser aplicado o método de *Hough* para deteção de linhas na imagem, foi necessário proceder a algumas modificações previas através da deteção de contornos [40]. Depois de aplicada a transformada de *Hough*, tornava-se mais fácil a determinação das coordenadas do ponteiro na imagem, possibilitando comparação com as de referência. De notar que ambos os métodos apresentam parâmetros de personalização que tiveram que ser modificados para os testes realizados.

## 3.5 - Sincronização dos relógios

Para que os sistemas permaneçam todos integrados, os comandos da unidade de comparação e processamento das imagens, bem como os comandos da unidade de captura das mesmas, teriam que estar sincronizados, daí existir a necessidade de sincronizar os relógios dos dois sistemas. Para que o sistema de comparação e processamento pudesse identificar qual das imagens teria que utilizar para a comparação, o dia, a hora, o minuto e o segundo que era apresentado nesta unidade, teria que ser exatamente o mesmo valor na unidade de aquisição.

Uma vez que o tempo de resposta por parte da aquisição, e o tempo de resposta por parte da interceção da imagem do painel de instrumentos, não são os mesmos, a sincronização de relógios permite dizer a estes dois sistemas que as ações pretendidas, vão ser executadas após um número segundos do sinal de envio. Este número é um valor que o utilizador define, garantindo assim, que ambos os sistemas executam as capturas ao mesmo tempo.

## 3.6 - Procedimento de aquisição de imagens

Após todos os sistemas estarem sincronizados, e quando estiverem prontos, o computador inicia a ordem de captura, enviando os comandos para a unidade de aquisição e para o painel de instrumentos. Ambos vão internamente, conforme as configurações previamente escolhidas, executar a sua função dando especial atenção à unidade de aquisição. Nesta unidade, uma vez que existem dois modos de captura, o modo vídeo e o modo imagem, a configuração de cada modo tem que ser previamente escolhida pelo utilizador. Os comandos de configuração vão ser tratados no subcapítulo seguinte, com mais detalhe.

Relativamente aos comandos de captura, existem dois modos de operação da *framework* de testes, o modo manual e o modo automático. Isto possibilita a utilização da ferramenta sem ser necessário testar ou capturar imagem dando a possibilidade de introduzir as imagens manualmente e fazer os testes automaticamente. O utilizador tem à sua disposição uma interface elementar baseada na consola do Linux, onde poderá selecionar entre os diferentes modos. No modo completamente automático, todos os comandos de captura efetuados, são introduzidos num ficheiro de texto de nome "*testes.txt*", que vão ser executados automaticamente, caso o modo automático tenha sido selecionado.

## 3.7 - Comandos de configuração

Os comandos de configuração devem ser executados sempre que necessários, antes dos comandos de captura. Isto pelos motivos referidos anteriormente, que a unidade de aquisição tem dois tipos de captura, o modo imagem e o modo vídeo. Caso o comando de configuração

não seja executado antes do comando de captura, o sistema vai assumir a última configuração introduzida, e desta forma poderá ou não executar o pretendido. Estes comandos de configuração aplicam-se tanto para o modo de funcionamento automático, como para o modo manual.

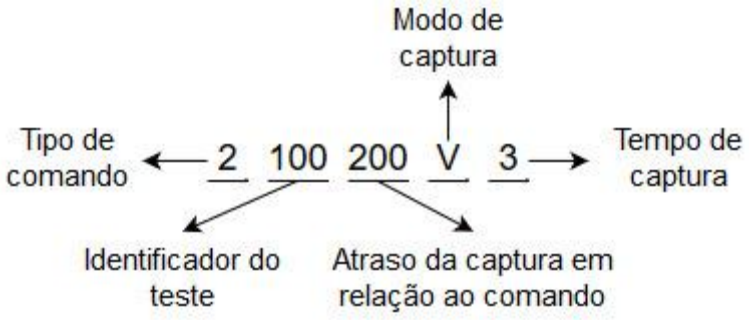



Figura 3.2 - Exemplo de um comando de configuração.

O comando de configuração da Figura 3.2 apresenta diversos parâmetros todos eles legendados. Por ordem, temos o número identificativo do tipo de comando a executar, neste caso de configuração, será o número '2', seguido de um número, de 3 dígitos, que corresponde a identificação do teste. Esta identificação tem que ser única de teste para teste, já que, todos os testes, tem como objetivo testar uma funcionalidade diferente no painel de instrumentos. Logo de seguida deve ser introduzido o atraso pretendido, para que a captura e a interceção dos sistemas correspondentes seja ao mesmo tempo. Por norma este valor deve estar compreendido em poucas centenas de milissegundos. O próximo parâmetro a ser introduzido deve ser o modo de captura, neste caso a letra 'I' para captura de uma única imagem, ou a letra 'V' para capturar vídeo. Caso a última seja selecionada, ainda existe um outro parâmetro a ser configurado que é a duração do vídeo. Este deve ser inserido em segundos, para que, mais tarde possa ser dividido em *frames* por segundo. A Figura 3.2 é um exemplo de um comando de configuração, que deve ser inserido no ficheiro de texto de nome "*comandos.txt*".

## 3.8 - Interface com o utilizador

Para uma melhor interação por parte do utilizador com a *framework* de testes, foi criada uma pequena e simples interface na linha de comandos que tem por objetivo uma melhor interação com o sistema. Na Figura 3.3 podemos ter uma perceção de como é possível interagir com a *framework* de testes e os vários estados da interface.

Depois de selecionar o modo manual, é apresentado ao utilizador um outro menu, que dá a opção de executar a captura, configurar os parâmetros da mesma, executar testes às imagens previamente capturadas, ou sincronizar os relógios dos sistemas.

O utilizador pode, à semelhança do modo de funcionamento, escolher um modo automático ou um modo manual, para a execução dos testes, sendo que neste último modo, o utilizador terá que selecionar que ficheiros deseja testar.

Para a sincronização de relógios, o utilizador deve selecionar essa opção, sendo assim enviado os respetivos comandos para os diferentes sistemas. Na *framework* de testes, após ser enviado o pedido para sincronização, é obtida uma resposta positiva caso a sincronização tenha corrido como esperado em todos os sistemas, ou negativa caso algo tenha falhado.

Seguindo uma sequência de parâmetros introduzidos na interface, teríamos o número 2 para o modo manual o número 3 para o modo de testes seguido do número 1 para o modo de testes automático. Na Figura 3.3 podemos ver esse exemplo realçado a vermelho. No anexo A está presente uma exemplificação da interface.

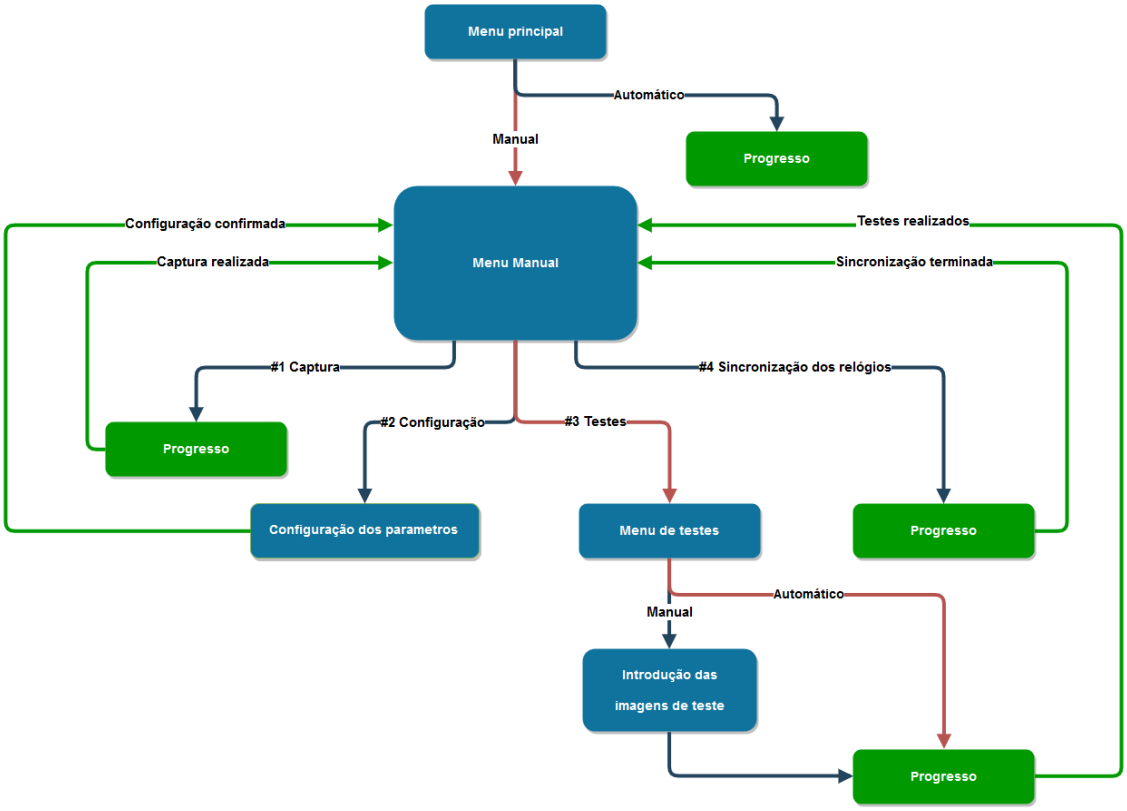



Figura 3.3 - Diagrama de alto nível da *framework* de testes. A azul estão representadas as ações que o utilizador pode interagir e a verde as que não pode. A vermelho encontramos o caminho para o exemplo mencionado em cima.

# Capítulo 4 – Aquisição das imagens de teste

## 4.1 - Hardware

Para a aquisição das imagens, foi utilizado como base uma *RaspberryPi*, como referido anteriormente. A *RaspberryPi*, é um minicomputador, desenvolvido no Reino Unido, pela fundação *Raspberry Pi*, em que todo o hardware desde processador às interfaces externas, são integradas numa única placa[41]. Neste projeto, o papel desta peça de hardware é fazer a captura através de uma câmara, do painel de instrumentos, para que essa imagem possa ser comparada também, com a imagem obtida através da FPGA.

A unidade de aquisição, também vai ter como função numa fase inicial, que é a de base de dados. Desta forma, evita-se uma transferência de ficheiros, e o custo de uma base de dados externa.

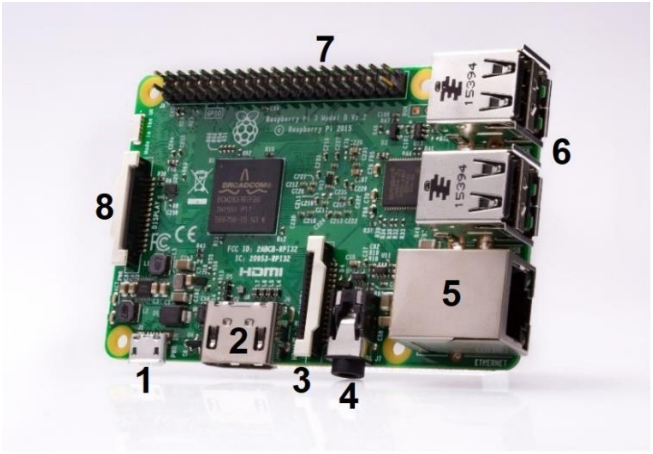



Figura 4.1 - 1 - *Power*; 2 - interface HDMI; 3 - interface câmara; 4 - interface áudio; 5 - *Ethernet*; 6 - portas *USB*; 7 - GPIO; 8 - interface *display*;[42].

Relativamente a constituição da *RaspberryPi*, a usada foi a versão 3 modelo B, ilustrada na Figura 4.1. A *RaspberryPi*, apresenta um processador *Quad Core 1.2GHz Broadcom 64 bit*, 1GB de *RAM*, suficiente para o trabalho a ser desenvolvido e uma placa de rede wireless que se mostrou importante neste projeto, e que mais tarde tornou-se muito relevante para uma maior portabilidade do sistema. Para que esta máquina funcione, existe uma entrada de

cartão de memória, onde deve conter o sistema operativo a ser usado. Na Figura 4.2 está representado um esquema que ilustra todas as comunicações com o exterior.

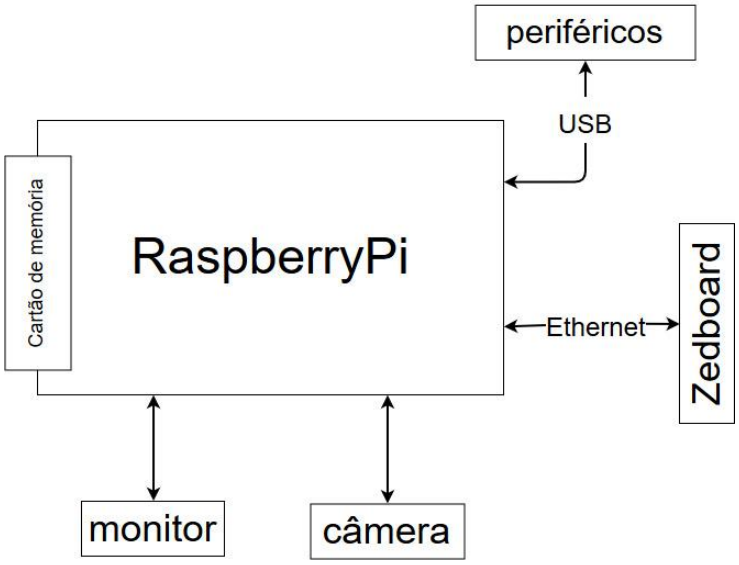



Figura 4.2 - Esquemático simplificado de todas as comunicações da *RaspberryPi* com o exterior neste projeto. Os periféricos neste diagrama correspondem ao rato e teclado usados para controlar a unidade.

## 4.2 - Código da unidade de aquisição de imagens

### 4.2.1 -Comandos de captura de imagem e vídeo

Para as capturas de teste, foi enviado para a câmara na unidade de aquisição o comando ***"raspistill -v -o test.png"***. O primeiro parâmetro é o modo de captura, para que a câmara identifique que é uma imagem que estamos a querer capturar. No fim temos o nome do ficheiro, que para este exemplo será "*test*" com o formato "*png*".

Também na mesma documentação, encontram-se os comandos necessários para fazer captura de vídeo, ***"raspvid -o vid.h264"***. À semelhança do comando de captura de imagens, a primeira palavra tem a ver com o modo de captura, e o parâmetro seguinte tem a ver com o nome e tipo de ficheiro de saída, que neste caso, assumia o nome de *"vid.h264"*. Este tipo de vídeo adiciona já alguma compressão que faz com que haja alguma perda de informação. Para a captura de vídeo, neste projeto, foram necessários testes mais exaustivos dada a variedade de parâmetros de configuração. Nas referências é possível encontrar o *link* para o website onde estes parâmetros foram retirados, podendo ser aí consultados todos os outros que não foram utilizados para este trabalho.

### 4.2.2 -Ligação com as outras unidades

Começou-se por implementar uma comunicação por *sockets*, com o protocolo TCP/IP, visto ser funcional e uma das formas mais fáceis de estabelecer comunicação entre o computador, onde estava a unidade de processamento, e a unidade de aquisição. Desta forma, o servidor estaria à espera de comunicações numa dada porta, por parte da unidade de processamento, funcionando assim como *slave* e a unidade de processamento como *master*. Caso houvesse um pedido de comunicação, o servidor criava um *socket* para aquela comunicação e atribuía aquele cliente ao respetivo *socket,* estando estabelecida a comunicação.

De seguida, a unidade de aquisição fica à espera de receber o parâmetro por parte da unidade de processamento, para executar uma das três ações possíveis, captura, sincronização dos relógios ou configuração dos parâmetros. Esta última, deverá ser efetuada pelo menos uma vez antes de dar início a uma captura.

No final das ações da unidade de aquisição, é enviado para a unidade de processamento uma confirmação de que foram todas executadas.

O mesmo tipo de comunicação foi implementado na unidade de aquisição depois de uma análise do código produzido nos trabalhos anteriores[18]. Mais uma vez a comunicação é feita por *sockets* assente no protocolo TCP/IP.

### 4.2.3 -Transferência de dados

A transferência de dados podia ser realizada através de *sockets* enviando a imagem pelo mesmo. Contudo, para além de haver tráfego de dados desnecessário, a imagem capturada e posteriormente enviada, apenas estaria disponível no computador onde estava a ser executada a *framework* de testes. Deste modo, optou-se por criar uma pasta partilhada, alojada na unidade de aquisição, que iria estar disponível em toda a rede wireless. Desta forma, torna-se possível a acessibilidade de qualquer computador na rede, não só a verificar a imagem capturada, mas também, a consultar os *logs* gerados. Para além desta enorme vantagem de poder consultar o processo de captura e teste em qualquer lugar, desde que na mesma rede da unidade de aquisição de imagens, também existe a vantagem de que não é necessário interromper os testes para ir verificando o processo, dada a consulta ser noutra máquina.

## 4.3 - Ambiente simulado

Dada a impossibilidade de utilizar o painel que se queria testar, uma vez que este apresentava uma imagem em branco, como se pode verificar no capítulo referente aos

resultados, optou-se por criar um ambiente simulado na unidade de aquisição de imagem para se poder validar os algoritmos de comparação. O ambiente consistia em apresentar uma imagem exemplo, num monitor, e assim fazer a captura através da *RaspberryPi*. A imagem utilizada para o ambiente simulado, pode-se ver na Figura 4.3, que foi a imagem mais clara e real que se conseguiu obter, de anos anteriores.

O ambiente simulado é um simples programa a ser executado na *RaspberryPi* onde, à semelhança do programa de servidor, apresenta uma comunicação por *sockets* no mesmo *ip*, mas, neste caso, com uma porta de rede diferente, permitindo a comunicação com os dois programas de forma independente. Também à semelhança do programa do servidor, o do ambiente simulado, após iniciado, fica à espera de um parâmetro, para mostrar no monitor a imagem. A abertura da mesma é feita com o programa de visualização de imagens da *RaspberryPi*, em modo *fullscreen* para eliminar eventual "ruído" externo.


Figura 4.3 - Imagem do painel funcional usada no ambiente simulado[18].

# Capítulo 5 – Resultados

Neste capítulo vão ser apresentados os resultados bem como uma reflexão dos mesmos conforme o que era esperado, e o que foi obtido. Na Figura 5.1 podemos observar o sistema completo pronto para testes finais.

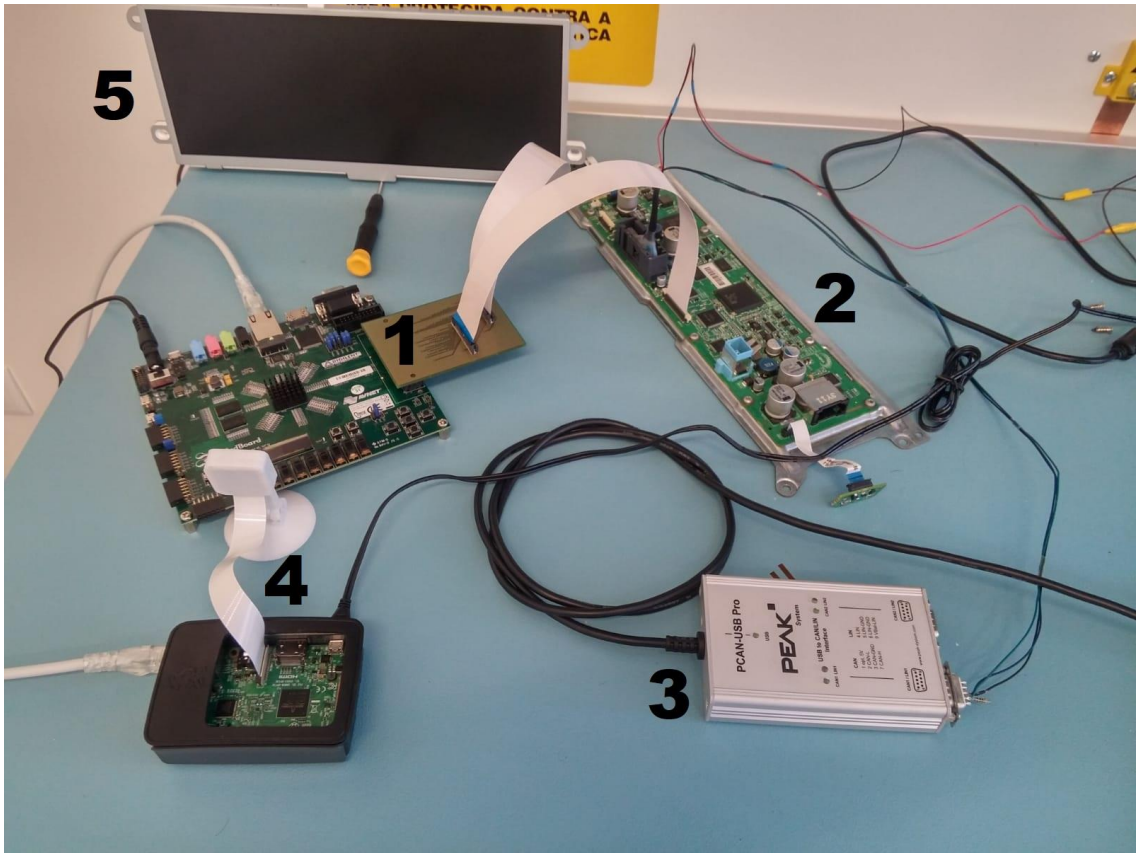



Figura 5.1 - Sistema completo para testes finais.

O sistema eletrónico do painel representado com o número 2 e o ecrã do painel com o número 5, estão a ser alimentados por uma fonte de tensão regulada para os 12V. Com o número 1 podemos observar a PCB projetada e produzida para estabelecer a ponte entre a *Zedboard* e o painel. Ligada a esta PCB está a *Zedboard* que se encontra conectada por *Ethernet* à *RaspberryPi*. Com o número 3 podemos observar a interface CAN/USB. Assinalado

com o número 4, temos a nossa unidade de aquisição de imagem em que podemos claramente observar ligado a branco o módulo da câmara direcionado para o painel.

Para ligar o painel de instrumentos, além de necessitar de 12V de tensão e um mínimo de 1A de corrente, também necessita do envio de uma série de comandos CAN para que seja visível qualquer informação no ecrã. Após várias tentativas de decifrar quais os comandos necessários para inicializar o *cluster*, dada a escassez de documentação, foi finalmente possível criar um conjunto de comandos, exemplificados no anexo D, que se concluiu serem os corretos para a inicialização, conforme os resultados apresentados e com base num documento fornecido pela empresa. Com o auxílio deste mesmo ficheiro, foi possível distinguir alguns comandos que seriam relevantes nomeadamente a variação da velocidade e das rotações, tendo estes comandos o ID 336 e 208 do anexo D. Desmontado o painel, na procura do problema para o mau funcionamento do mesmo, não foi possível encontrar qualquer anomalia. Os resultados da inicialização podem ser visíveis na Figura 5.2.

Figura 5.2 - Resultado dos comandos de inicialização. De notar que o painel apresenta luminosidade e o sinal do "airbag" ligado durante alguns segundos, indicador do processo de incialização.

Começando por identificar novamente os requisitos, o sistema tem que ser capaz de capturar imagens; reconhecer nessas imagens símbolos e números; fazer a calibração da imagem caso o painel não esteja alinhado; fazer reconhecimento de posições dos ponteiros no painel ao longo do tempo; guardar imagem de referência caso não haja nenhuma imagem para comparar.

## 5.1 - Unidade de processamento

À medida que foi desenvolvido a *framework* de testes, foram feitos testes aos algoritmos usados, tal como é referido na metodologia adotada. Foi então utilizada a imagem de teste

adquirida nos anos anteriores para efeitos de reconhecimento de símbolos e números. Utilizou-se seguidamente uma imagem exemplo do símbolo a testar e aplicou-se o método implementado. O resultado foi bastante positivo, tendo sido reconhecido o símbolo como se pode verificar na Figura 5.3. Nesta figura já se encontra implementado o método de calibração, sendo possível observar uma pequena rotação no sentido dos ponteiros do relógio para alinhar os dois mostradores, da velocidade e das rotações do automóvel. Relativamente aos números, não foi possível obter números de quilometragem presente no painel. No entanto, foi possível o reconhecimento de um número do velocímetro, tendo sido então validada a forma como o reconhecimento é feito. Podemos observar os resultados na Figura 5.4.


Figura 5.3 - Reconhecimento com sucesso do símbolo de perigo na imagem de teste.


Figura 5.4 - Reconhecimento com sucesso do número 10 na imagem de teste. É possível verificar que a imagem de teste não apresentava valores na zona de distância percorrida e como tal não foi possível testar esses números.

Faltava testar o modo *Sweep* mas uma vez que o painel não apresentava qualquer imagem válida, este não foi possível ser completamente testado, e como tal, teve que se usar a imagem de referência. Para simular a variação do ponteiro da velocidade, esta imagem foi alterada para que pudesse ser identificado o ponteiro em várias posições. O resultado está presente na Figura 5.5 e na Figura 5.6.

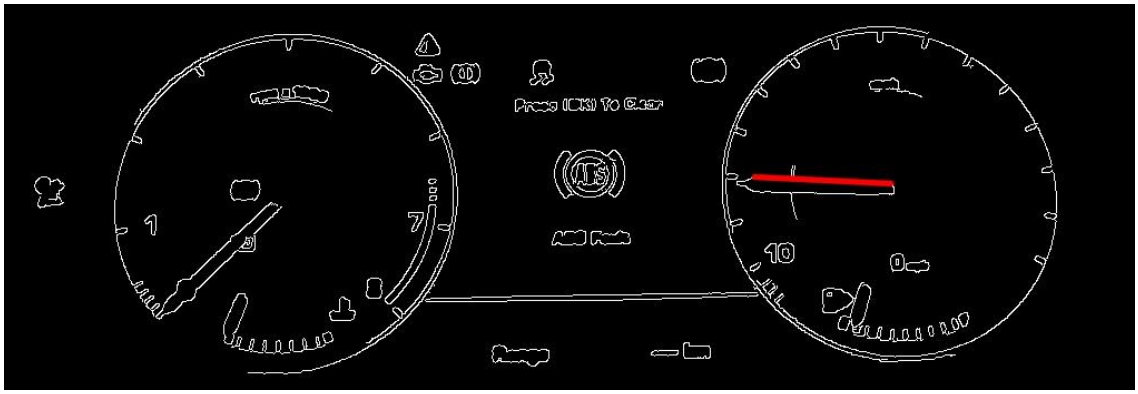


Figura 5.5 - Reconhecimento com sucesso do ponteiro do velocímetro sendo de seguida comparada a posição com a posição de referência.

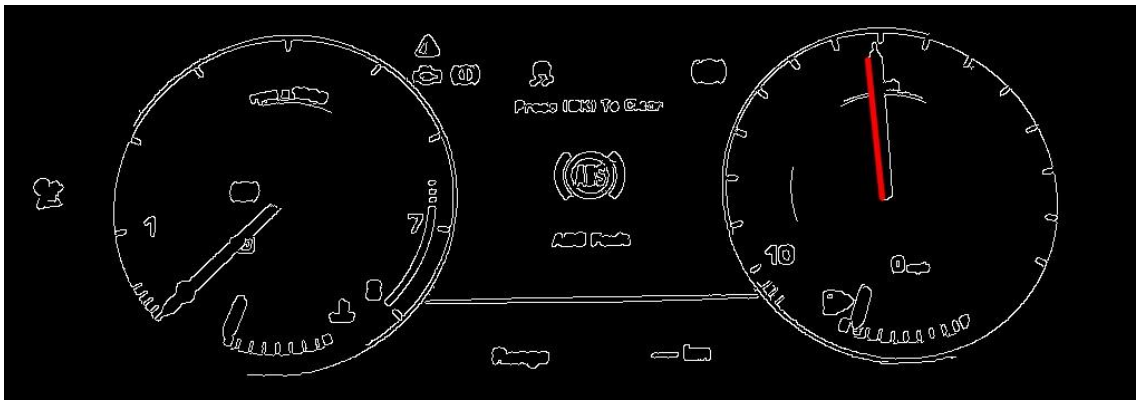


Figura 5.6 - Reconhecimento com sucesso do ponteiro do velocímetro. A imagem inicial foi alterada para que o ponteiro fosse alterado da posição da imagem original para a posição que se pode observar.

Para se analisar quanto tempo demoraria um painel a ser testado, foram feitos testes de performance. Para os testes, foram usadas 35 capturas de imagens estáticas e 4 vídeos, 2 deles com a duração de 3 segundos, simulando assim o tempo que os ponteiros maiores demorariam a percorrer a escala completa, e os outros 2 vídeos de 2 segundos simulando o tempo dos ponteiros da temperatura e combustível, perfazendo um total de 650 *frames*, admitindo uma média de 60 *frames* por segundo na aquisição dos vídeos.

Estes valores foram baseados no tamanho do ecrã e as regiões onde poderiam aparecer números e símbolos. Além disso. para que se pudesse obter um tempo real aproximado da validação completa do painel, foram simulados 50 testes constituídos cada um pelas 650 *frames* capturadas e comparadas, para se poder calcular o desvio padrão e a média desses resultados. A quantidade de testes realizados deveu-se não só à falta de tempo para a realização de mais, mas também porque um maior número não faria variar muito os resultados obtidos.

Também foram realizados testes de performance a cada um dos testes aplicados, o de calibração, o de *sweep* e o de reconhecimento de símbolos. Como termo de comparação,

testaram-se dois símbolos diferentes, o de ABS, presente no centro do ecrã do painel e o do sinal de perigo que se pode visualizar no canto superior esquerdo na zona central do painel.


Figura 5.7 – Imagem de pesquisa dos números referidos anteriormente estando representado com o número 1 o símbolo de ABS e com o número 2 o símbolo de perigo.

Foram realizadas 650 vezes cada um dos testes. Os resultados podem-se observar na Tabela 5.1, sendo que o símbolo de ABS foi, em média, mais rápido de encontrar, como era esperado, em cerca de 12 milissegundos face ao símbolo de perigo.

Tabela 5.1 - Resultados dos tempos em milissegundos de execução do reconhecimento dos símbolos.

| | **Média (ms)** | **Desvio-Padrão(ms)** | **Dimensões *template*(px)** |
|---|---|---|---|
| **Símbolo Perigo** | 97 | 3.4 | 33x22 |
| **Símbolo ABS** | 108 | 7.3 | 83x54 |

Para os testes de calibração, à semelhança dos testes anteriores foram realizados 650 exemplos tendo sido calculados os resultados apresentados na Tabela 5.2.

Tabela 5.2 - Resultados dos tempos em milissegundos de execução dos testes de calibração e *Sweep*.

| | **Média (ms)** | **Desvio-Padrão(ms)** |
|---|---|---|
| **Calibração** | 249 | 8.1 |
| **Teste sweep** | 41 | 1.1 |

Uma análise crítica dos valores, conclui-se que para reduzir estes tempos de teste, principalmente nas comparações, seria relevante em vez do símbolo ou número estar a ser pesquisado na imagem completa, delimitar uma região previamente definida onde a pesquisa fosse mais rápida. Foi então utilizada uma região dando origem a uma nova imagem com um

tamanho de 676x252 pixéis em vez do tamanho original da imagem que era 1176x696 pixéis. Outra forma de reduzir o tempo de aquisição, seria obviamente reduzir o número de aquisições realizadas e com uma aquisição, poder ser testado múltiplos símbolos ou números. Os tempos aproximados sem e com região podem ser contemplados na Tabela 5.3. De notar que a região foi aplicada nos testes e não na aquisição. A variação relativamente à aquisição tem origem nos valores obtidos aquando da realização dos mesmos. Todos os valores obtidos dependeram das características da unidade de processamento, e uma vez que esta unidade estava a ser usada para outros fins, ao mesmo tempo que os testes estavam a ser executados, estes valores podem não corresponder aos tempos ótimos do processo de validação. As especificações da máquina podem ser observadas na Tabela 5.4.

Tabela 5.3 - Tempos de performance em segundos dos resultados obtidos.

| | **Média Aquisição (s)** | **Média Testes (s)** | **Total (s)** |
|---|---|---|---|
| **Sem região** | 702 | 860 | 1563 |
| **Com região** | 701 | 748 | 1448 |
| **Desvio Padrão** | 12 | 21 | - |

Tabela 5.4 - Especificações técnicas do computador usado para os testes de performance.

| **CPU** | **CPU Intel® Core™i7-6700HQ @2.60GHz (4 cores, 8 threads)** |
|---|---|
| **Memoria RAM** | 16GB 2400MHz DDR4 |
| **Sistema Operativo** | Windows 10 Pro 64 bits |
| **Disco** | SSD Samsung PM981 SSD NVMe PCIe M.2 512GB |

## 5.2 - Unidade de interceção

Depois da unidade de processamento testada, foi necessário implementar algumas alterações na unidade de interceção para que ficasse pronta a receber as informações do painel.

Inicialmente foi necessário atualizar todos os "IP cores" desatualizados. De seguida, foram então acrescentadas portas como saídas, tal como podemos ver na Figura 5.8, e ligadas aos sinais de entrada vindos do painel. Não existe nenhuma alteração dos sinais ao longo da

"*datapath*" do projeto, fazendo com que todos os sinais recebidos sejam enviados para o ecrã do painel de instrumentos, através da PCB projetada.

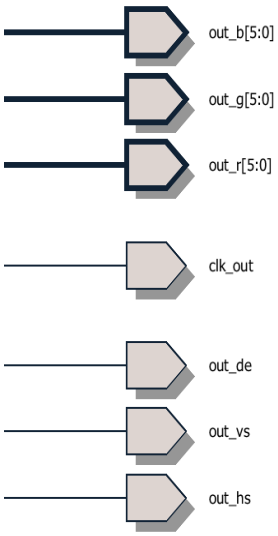



Figura 5.8 - Portas adicionados ao "*Datapath*" do projeto.

Por fim alterou-se o ficheiro das restrições, para estabelecer a ligação entre as portas internas acrescentadas anteriormente e as ligações físicas. Para tal recorreu-se a um documento que fazia a ligação entre as variáveis internas e os terminais da ficha FMC[42]. Os terminais da ficha FMC foram escolhidos para tornar mais claro e organizado o layout da PCB, possibilitando uma análise da mesma, muito mais fácil. Esta pode-se observar com mais detalhe na Figura 5.9 e na Figura 5.10.

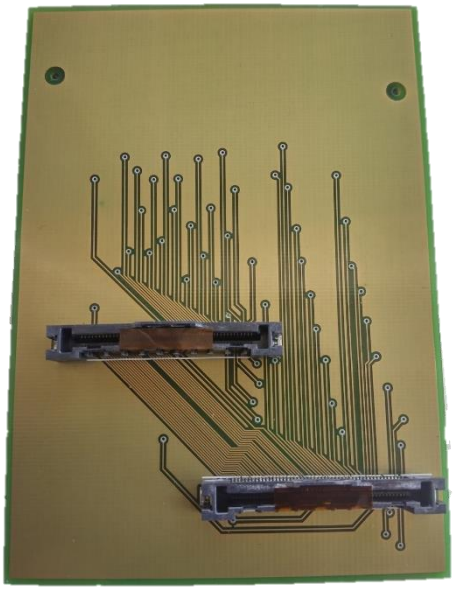


Figura 5.9 – PCB real vista de cima

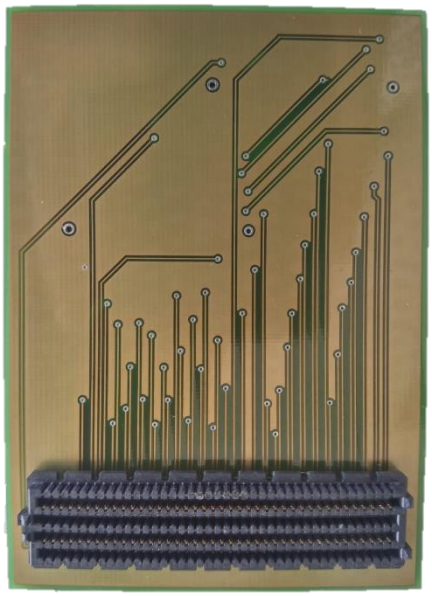


Figura 5.10 - PCB real vista de baixo

# Capítulo 6 – Conclusões E trabalho futuro

Neste capítulo, podemos encontrar uma breve reflexão de todo o trabalho pesquisado, realizado e futuro.

Marcas de automóveis, usuárias de painéis completamente digitais nos seus carros, já apresentam, uma validação destes painéis, de uma forma muito semelhante à realizada neste projeto. Esta informação foi obtida por um dos principais seguidores e contribuintes deste projeto, o Eng. Sénior Luís Cruz da CRITICAL Software, que teve oportunidade de obter essa informação junto de uma das pessoas responsáveis por essas validações. O sistema que apresentam, baseia-se num conjunto de algumas câmaras, posicionadas estrategicamente para capturarem o painel a validar. Estas imagens são usadas para fazer o reconhecimento de símbolos, números, entre outros fatores. Existe uma diferença significativa em relação a este projeto relativamente ao custo do equipamento usado, uma vez que é muito mais dispendioso em relação ao projeto atual. Ainda assim, como normalmente é de esperar, quanto mais caro o equipamento maior a qualidade do mesmo, logo, melhores os resultados que estes vão apresentar. Também este sistema apresenta uma componente completamente automática deixando livre, face ao sistema atual de validação, um recurso humano essencial noutras atividades. Conclui-se assim, que o sistema realizado no âmbito desta dissertação, tem potencial para se tornar num sistema fiável e modular, a fim de poder ser usado noutros painéis de outros veículos, uma vez que parte da mesma base que outros equipamentos funcionais e operacionais.

A presente dissertação surgiu da necessidade de automatizar os testes e a validação dos painéis de instrumentos completamente digitais dos automóveis. Desenvolvido no âmbito empresarial, tendo a CRITICAL Software sido a empresa criadora deste projeto, esta validação não ficaria apenas restrita a painéis digitais de automóveis. Sendo a CRITICAL Software uma empresa de desenvolvimento de software que visa responder à diversidade de outros sectores para além do automóvel, nomeadamente, o sector dos transportes ferroviários e aeroespaciais, é fundamental a modularidade do projeto para que possa vir a ser aplicado no âmbito dessas mesmas áreas.

O desafio proposto inicialmente por esta empresa, foi o da criação de um sistema capaz de autonomamente e com o mínimo de intervenção humana, pudesse testar o painel automóvel, do ponto de vista de resposta a comandos visuais. Além disso o projeto deveria

apresentar o menor custo possível, dada a existência de sistemas de validação muito mais sofisticados e consequentemente muito mais dispendiosos. Estes objetivos foram cumpridos, apesar de terem existido dificuldades relativamente aos componentes e softwares descritos anteriormente. Logo no início do projeto, foram sentidas dificuldades na análise de todo o sistema existente, em consequência da falta de bases na área de *FPGA's*. Após a compreensão desse sistema, instalou-se a biblioteca *OpenCV*, que por diversos motivos não foi possível ser instalada imediatamente, tendo sido necessário um trabalho extra para a resolução destas dificuldades. Toda a logística de montagem do sistema e configuração do mesmo sempre que era necessário torná-lo novamente operacional para realizar testes, consumiu bastante tempo e trabalho. Ao longo deste projeto, a falta de recursos humanos e suporte técnico, levou que o processo de análise e construção do software fosse mais lenta. Ainda assim, todas estas dificuldades foram superadas, o sistema desenvolvido, composto por uma fase de comandos, uma fase de aquisição da imagem, e uma fase de comparação da imagem captura com a imagem de referência, foi bem-sucedido. A modularidade do sistema, foi pensada ao longo do desenvolvimento do mesmo, concluindo que era uma mais valia para o projeto, uma vez que seria possível, em fase de testes o painel ficar num ambiente controlado a nível de luz, e o utilizador da ferramenta desenvolvida pudesse estar noutro local assistindo a todos os testes, e à validação, desde que estivesse na mesma rede configurada pela *framework* de testes.

De salientar que todo o trabalho desenvolvido foi realizado com base nas necessidades da empresa tendo como ponte de partida trabalhos desenvolvidos anteriormente, relativamente a este grande projeto.

## 6.1 - Trabalho Futuro

Este projeto teve um avanço enorme desde os últimos realizados no ano de 2016, no entanto ainda não ficou concluído existindo sempre melhorias a realizar. Este subcapítulo destina-se a enumerar algumas dessas melhorias e explicar de que forma estas contribuiriam para um enriquecimento do sistema. Para que todas estas melhorias pudessem ser realizadas necessitar-se-ia de mais tempo destinado à realização do projeto.

O painel não esteve operacional, pelo que não foi possível o teste completo com imagens vindas do mesmo, todavia seria bastante pertinente a utilização de um outro painel funcional e adaptação dos programas para que pudessem ser testados para o novo equipamento.

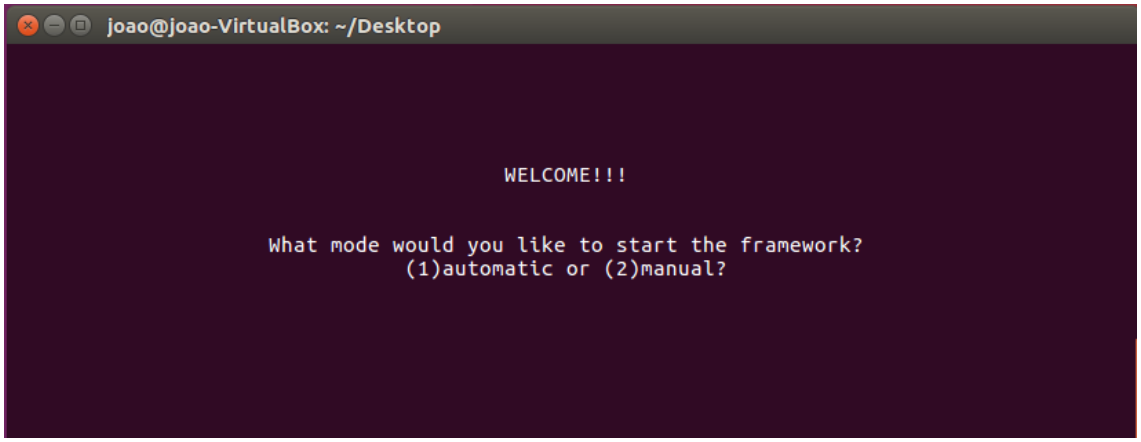
A existência de uma unidade de aquisição independente de uma unidade de comparação, torna o sistema por módulos e portátil. Contudo, a criação de uma vertente menos modular que eliminava a unidade de aquisição e a juntava com a unidade de processamento, tornaria o sistema ainda mais rápido e para casos muito específicos poderia ser vantajoso.

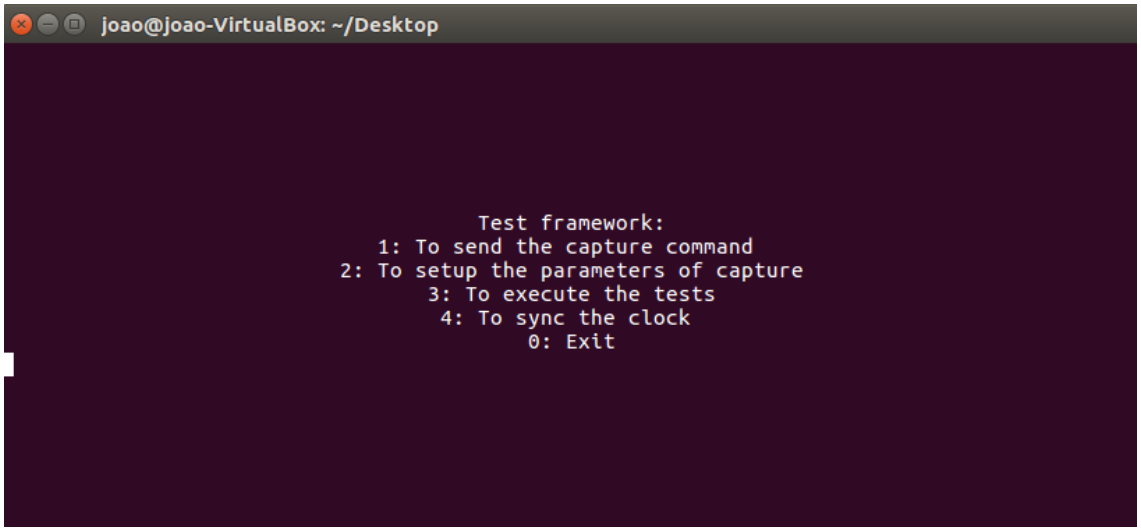
Olhando para o sistema como um todo, fica para trabalho futuro a integração completa da unidade de interceção da imagem vinda do painel, uma vez que também não foi possível testar com o painel atual. Ao realizar este teste automaticamente, iria ser realizado o teste à placa projetada e desenvolvida para que esta também pudesse ser validada.

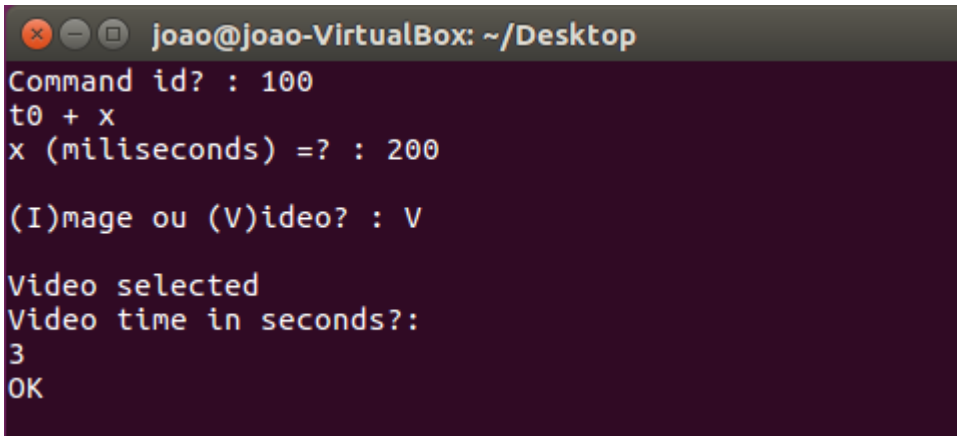
A nível de código, seria interessante o aperfeiçoamento de algumas funções e recursos de memória usados, já que o presente trabalho não o exigia.

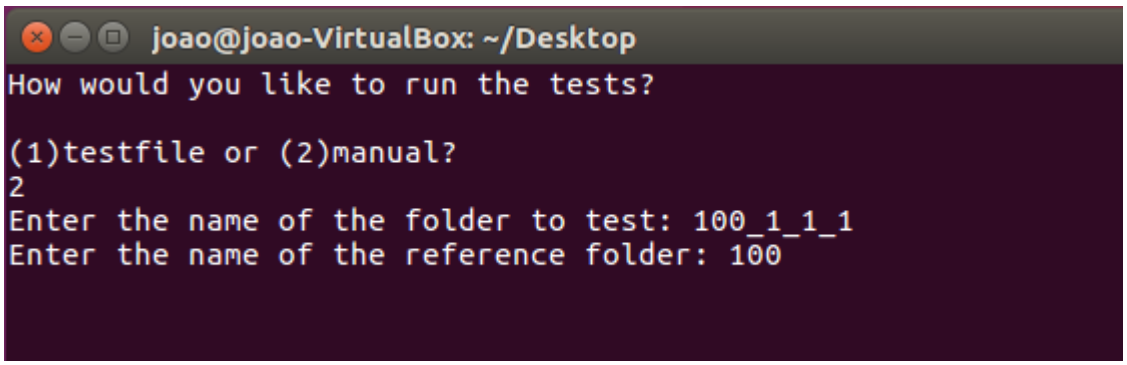
# Anexos

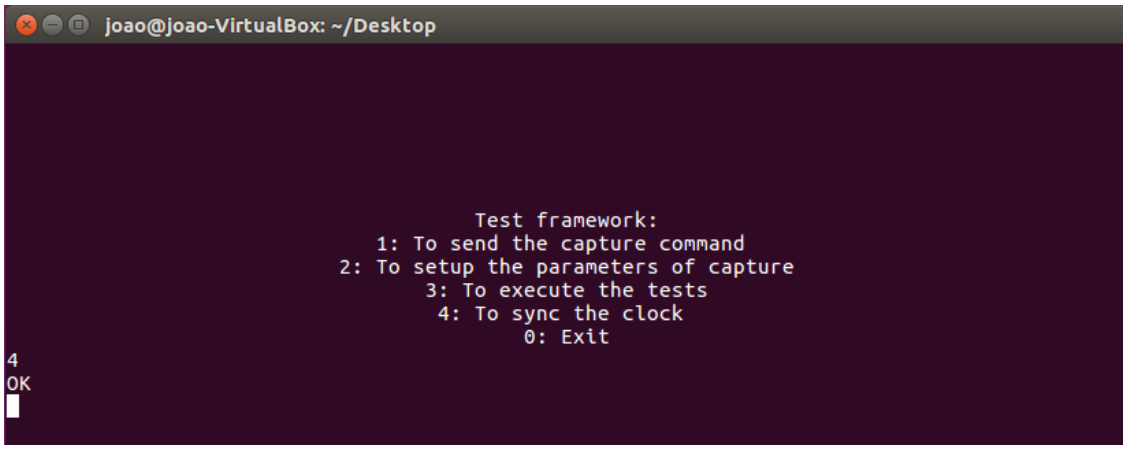
## Anexo A:  Menu do utilizador



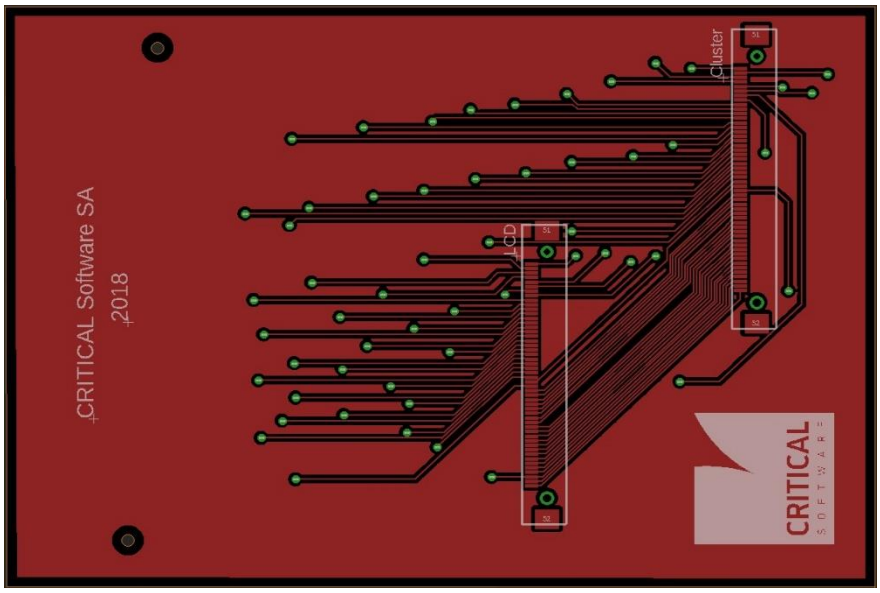



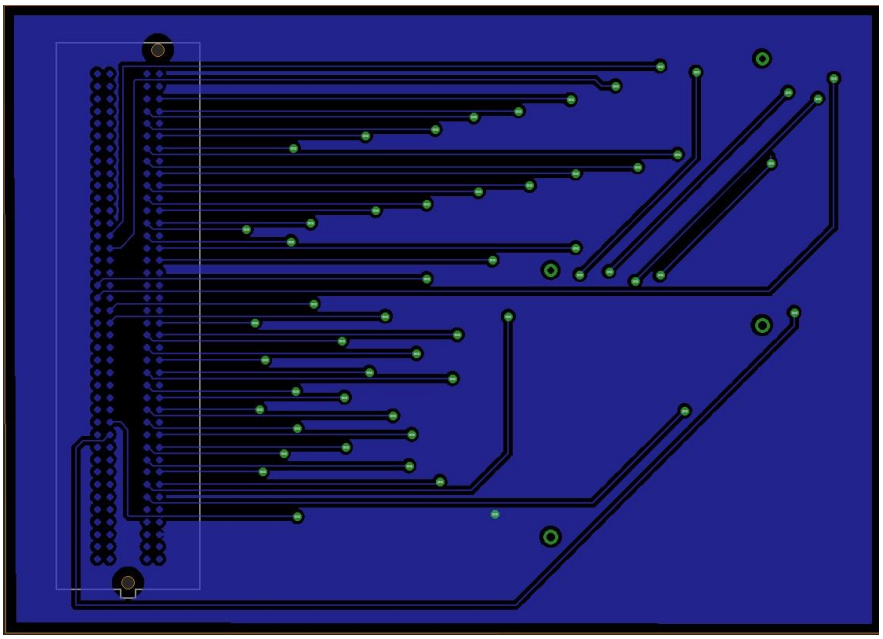

joao@joao-VirtualBox: ~/Desktop
Command id? : 100
t0 + x
x (miliseconds) =? : 200
(I)mage ou (V)ideo? : V
Video selected
Video time in seconds?:
3
OK


joao@joao-VirtualBox: ~/Desktop
How would you like to run the tests?
(1)testfile or (2)manual?
2
Enter the name of the folder to test: 100_1_1_1
Enter the name of the reference folder: 100

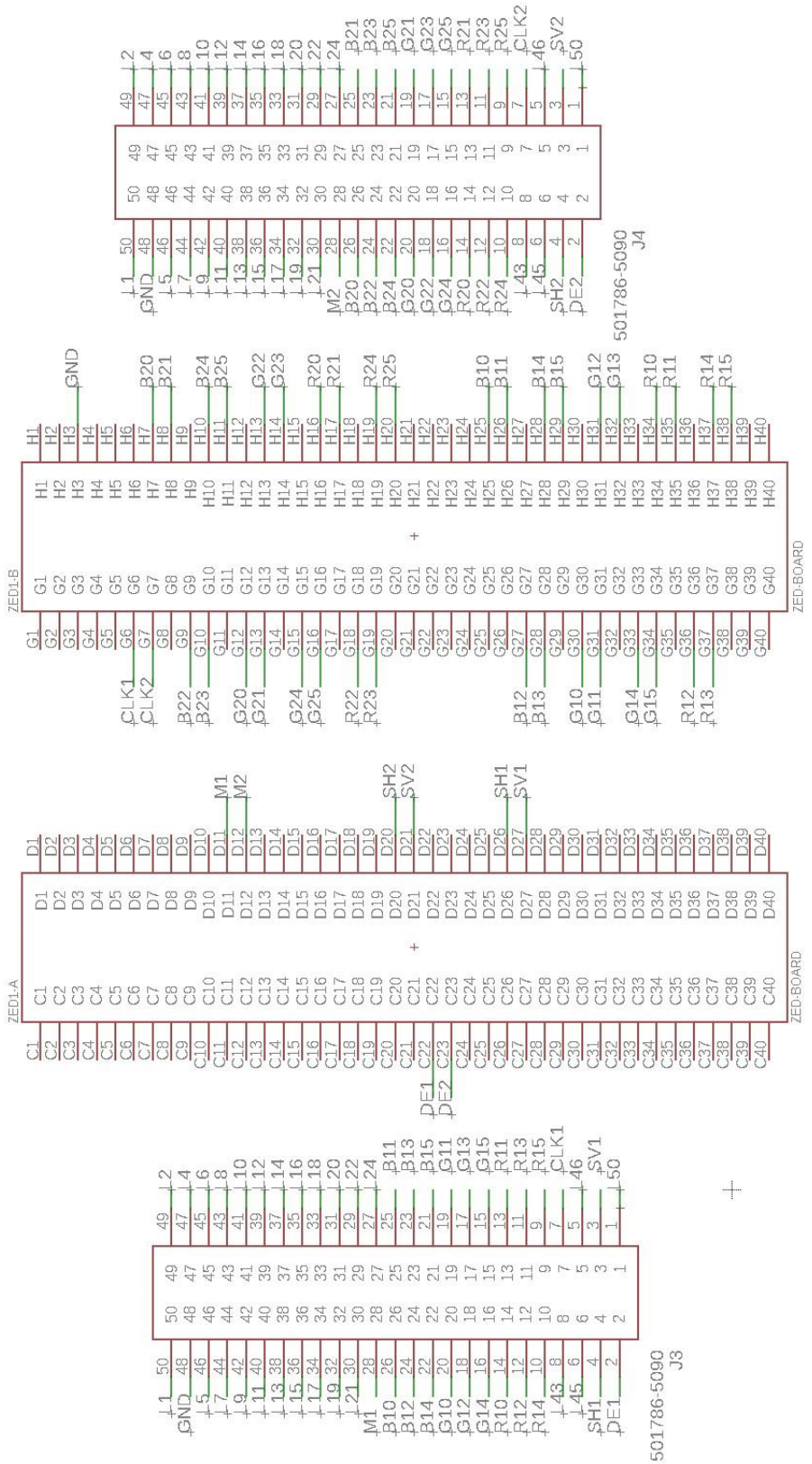

joao@joao-VirtualBox: ~/Desktop
Test framework:
1: To send the capture command
2: To setup the parameters of capture
3: To execute the tests
4: To sync the clock
0: Exit
4
OK

# Anexo B1: PCB Projetada

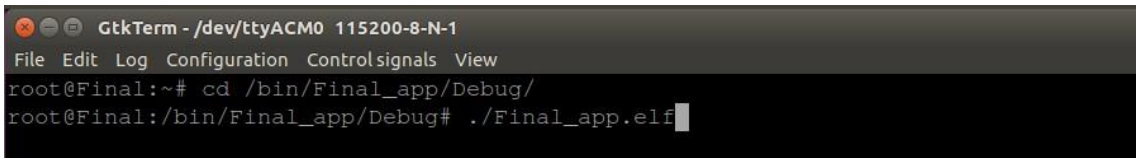


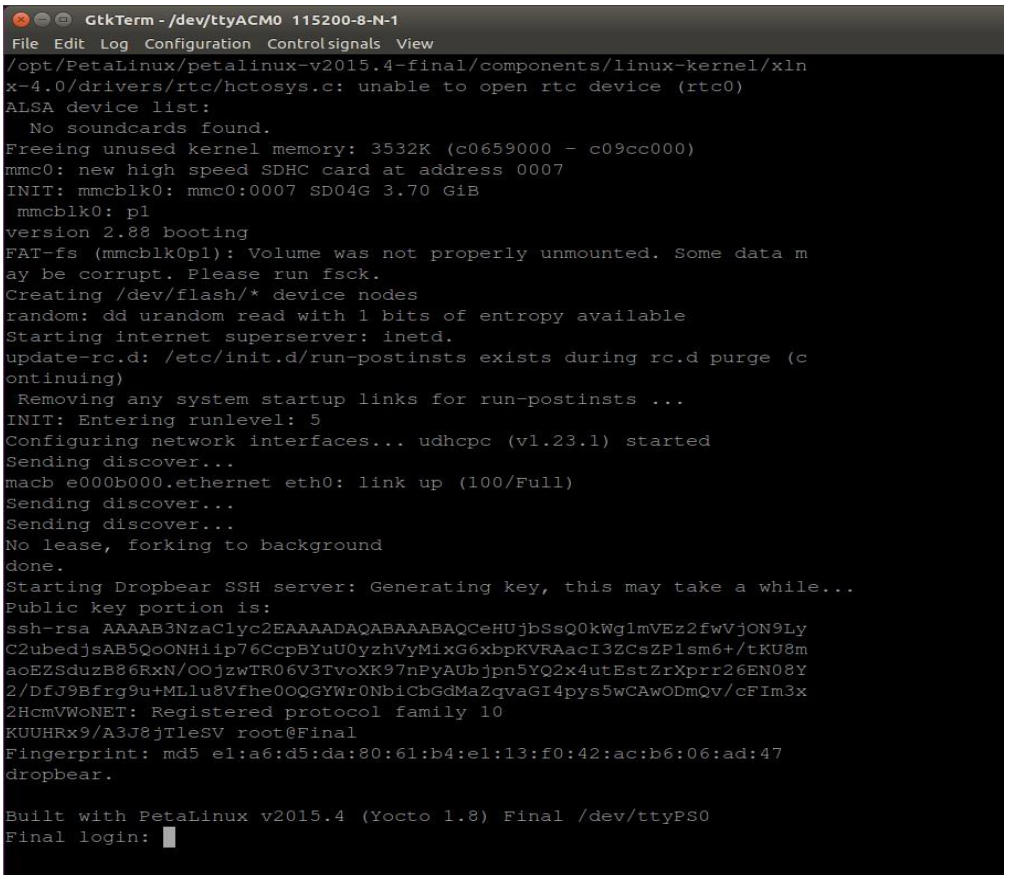



# Anexo B2: Esquemático PCB Projetada



# Anexo C: Inicialização do Petalinux



Anexo D1: Comandos de inicialização do painel de instrumentos.

| | CAN-ID | Type | Length | Data | Cycle Time |
|---|---|---|---|---|---|
| Transmit | 512 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 20 |
| | 320 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 50 |
| | 272 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 50 |
| | 336 | | 8 | 00 00 00 0A 76 00 00 30 | ☑ 50 |
| | 480 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 50 |
| | 656 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 50 |
| | 792 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 50 |
| | 812 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 736 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 896 | | 8 | A0 00 00 00 00 00 00 00 | ☑ 180 |
| | 608 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 872 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 880 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 192 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 400 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 208 | | 8 | C0 00 00 00 00 24 00 00 | ☑ 200 |
| | 560 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 816 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 688 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 200 |
| | 704 | | 8 | 00 00 00 00 00 00 A6 00 | ☑ 200 |
| | 1048 | | 8 | 00 00 00 0A C8 64 00 00 | ☑ 200 |
| | 864 | | 8 | 20 00 00 00 00 00 00 04 | ☑ 200 |
| | 848 | | 8 | 00 00 00 00 00 00 00 06 | ☑ 50 |
| | 384 | | 8 | 00 00 00 00 00 40 00 00 | ☑ 200 |
| | 1056 | | 8 | 00 00 20 00 00 00 00 04 | ☑ 200 |
| | 856 | | 8 | 00 00 00 00 00 03 00 04 | ☑ 200 |
| | 1052 | | 8 | 00 00 12 04 11 0A 09 00 | ☑ 200 |
| | 224 | | 8 | 00 00 00 00 1C 00 00 00 | ☑ 100 |
| | 592 | | 8 | 00 00 00 30 03 00 00 00 | ☑ 200 |
| | 768 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 100 |
| | 844 | | 8 | 01 43 53 20 CF E8 CB E8 | ☑ 200 |
| | 432 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 300 |
| | 352 | | 8 | 00 00 00 00 00 00 00 00 | ☑ 300 |
| | 1297 | | 8 | 11 02 00 00 00 00 00 00 | ☑ 1000 |
| | 1080 | | 8 | 3E 01 84 00 00 0D 8A F4 | ☑ 700 |

Anexo D2: Comandos recebidos do painel após inicialização.

Receive

| CAN-ID | Type | Length | Data | Cycle Time |
|---|---|---|---|---|
| 512 | | 8 | 00 00 00 00 00 00 00 00 | 24,2 |
| 768 | | 8 | 00 00 00 00 00 00 00 00 | 80,3 |
| 896 | | 8 | A0 00 00 00 00 00 00 00 | 179,4 |
| 1080 | | 8 | 40 01 84 00 00 0D 8A F4 | 731,8 |
| 1297 | | 8 | 11 02 00 00 00 00 00 00 | 192,4 |
| 844 | | 8 | 09 F1 FE F3 E8 C8 00 00 | 184,9 |

# Referências


[1] "Painel de instrumentos digital da Audi chega ao A3 em 2016," 2015. [Online]. Available: http://revistacarros.sapo.pt/painel-de-instrumentos-digital-da-audi-chega-ao-a3-em-2016/.

[2] B. Howard, "Digital dashboard: Why your car's next instrument panel will be one big LCD," 2012. [Online]. Available: https://www.extremetech.com/extreme/131485-digital-dashboard-why-your-cars-next-instrument-panel-will-be-one-big-lcd.

[3] J. Laukkonen, "Introducing the Instrument Cluster." [Online]. Available: http://www.crankshift.com/instrument-cluster/.

[4] V. V Alexandrov and N. D. Gorsky, "Can a Computer Vision System Work Like the Human One ?," vol. 3, pp. 269–277, 1991.

[5] A. Hudson Davies, "The Physical And Mental Effects Of Monotony In Modern Industry," vol. 2, no. 3427, pp. 472–479, 2013.

[6] K. Schwaber, *Agile Project Management with Scrum*, vol. 7, no. Cmm. 2004.

[7] K. Schwaber, "SCRUM Development Process," no. February 1986, pp. 1994–1995, 1997.

[8] Globalteckz.com, "Characteristics of Agile Methodology in Software Development," 2013. [Online]. Available: http://blogs.globalteckz.com/characteristics-of-agile-methodology-in-software-development/.

[9] C. Martins, "Como a Audi testa os sistemas eléctricos dos novos carros," 2014. [Online]. Available: https://abertoatedemadrugada.com/2014/05/como-audi-testa-os-sistemas-electricos.html.

[10] Y. Huang, A. Mouzakitis, R. McMurran, G. Dhadyalla, and R. P. Jones, "Design validation testing of vehicle instrument cluster using machine vision and hardware-in-the-loop," *Proc. 2008 IEEE Int. Conf. Veh. Electron. Safety, ICVES 2008*, pp. 265–270, 2008.

[11] J. P. A. Duarte, "Aquisição de Imagens através de FPGA," 2015.

[12] L.-B. Fredrikson, "CONTROLLER AREA NETWORKS AND TIIE PROTOCOL CAN FOR MACHINE CONTROL SYSTEMS," *Group*, vol. 4, no. 94.

[13] "Protocolo de comunicação CAN," pp. 37–61, 1986.

[14] "PCan USB Pro." [Online]. Available: https://www.peak-system.com/PCAN-USB-

Pro.200.0.html?&L=1.

[15] Y. Huang *et al.*, “Development of an automated testing system for vehicle infotainment system,” *Int. J. Adv. Manuf. Technol.*, vol. 51, no. 1–4, pp. 233–246, 2010.

[16] J. Johnson, “Comparison of Zynq boards,” 2014. [Online]. Available: http://www.fpgadeveloper.com/2014/03/comparison-of-zynq-boards.html.

[17] J. Zhao, “Video / Image Processing on FPGA,” no. April, 2015.

[18] P. N. de Q. S. de C. Geraldes, “AQUISIÇÃO DE IMAGENS ATRAVÉS DE FPGA,” 2016.

[19] ZedBoard, “ZedBoard Technical Specifications.” [Online]. Available: http://zedboard.org/content/zedboard-0.

[20] V. Gökmen, H. Z. Senyuva, B. Dülek, and E. Çetin, “Computer vision based analysis of potato chips – A tool for rapid detection of acrylamide level,” *Mol. Nutr. Food Res.*, vol. 50, no. 9, pp. 805–810, 2006.

[21] R. Szeliski, “Computer vision : algorithms and applications,” p. 812, 2011.

[22] D. Lowe, “The computer vision Industry,” 2015. [Online]. Available: http://www.cs.ubc.ca/~lowe/vision.html.

[23] G. P. Stein, E. Rushinek, G. Hayun, and A. Shashua, “A Computer Vision System on a Chip: a case study from the automotive domain,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. - Work.*, vol. 3, pp. 130–130, 2005.

[24] “Global Computer Vision Industry Research Analysis Expected to reach CAGR of 8.2% in Upcoming Years by 2025 With top key vendors like Facebook Inc, Google LLC, Microsoft Corporation , Nvidia Corporation, Mercedes-Benz , IBM Corporation.” [Online]. Available: https://www.openpr.com/news/1021723/Global-Computer-Vision-Industry-Research-Analysis-Expected-to-reach-CAGR-of-8-2-in-Upcoming-Years-by-2025-With-top-key-vendors-like-Facebook-Inc-Google-LLC-Microsoft-Corporation-Nvidia-Corporation-Mercedes-Benz-IBM-Corp.

[25] M. S. Corson and R. H. Moss, “Computer Vision - An in-depth look at technique and applications of digital image processing,” no. December, pp. 31–34, 1986.

[26] A. Dolgova, “VisionHack 2017: The State Of Computer Vision In Russia,” 2017. [Online]. Available: http://yonah.org/channel/visionhack-computer-vision-russia/.

[27] S. Patel and M. Goswami, “Comparative analysis of Histogram Equalization techniques,” *Proc. 2014 Int. Conf. Contemp. Comput. Informatics, IC3I 2014*, vol. 1, no. 2, pp. 167–168, 2014.

[28] P. Schelkens, A. Skodras, and T. Ebrahimi, *The JPEG 2000 Suite*. 2009.

[29] I. Richardson, *The H. 264 advanced video compression standard*. 2011.

[30] P. E. Hart, “How the Hough transform was invented [DSP History],” *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 18–22, 2009.

[31] S. Huijie and C. Qiang, “Computer vision recognition of incomplete symbols in Russian symbols,” vol. 653, pp. 2283–2286, 2014.

[32] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and

curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[33] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover.," *tech. Rep. C.*, p. 175, 1980.

[34] "Detection With Opencv," 2011. [Online]. Available: https://glowingpython.blogspot.com/2011/10/corner-detection-with-opencv.html.

[35] W. Burger and M. J. Burge, *Undergraduate Topics in Computer Science*. 2009.

[36] Opencv, "Template Matching." [Online]. Available: https://docs.opencv.org/trunk/d4/dc6/tutorial_py_template_matching.html.

[37] Z. Chaczko, L. a. Yeoh, and V. Mahadevan, "A preliminary investigation on computer vision for telemedicine systems using OpenCV," *ICMLC 2010 - 2nd Int. Conf. Mach. Learn. Comput.*, pp. 42–46, 2010.

[38] "OpenCV Website." [Online]. Available: https://opencv.org/about.html.

[39] "OpenCV - Match Template." [Online]. Available: https://docs.opencv.org/2.4.13.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html.

[40] J. Canny, "A computational approach to edge detection.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[41] "Raspberry PI v3 - Specifications." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

[42] Digilentinc, "fpga_package_pin_to_fmc." [Online]. Available: https://reference.digilentinc.com/_media/reference/programmable-logic/zedboard/fpga_package_pin_to_fmc.pdf.